IBM

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

- Understand the benefits of operational decision management
- Build dynamic solutions with business events and business rules
- Learn by example with practical scenarios

Janet Wall
Chris Backhouse
Sebastien Brunot
Pierre Feillet
Mark Hiscock
Mike Johnson
Yongxin Pan
Wang Wenjie
Timothy Wuthenow

Redbooks

ibm.com/redbooks

International Technical Support Organization

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

April 2012

SG24-8014-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (April 2012)

This edition applies to Version 7, Release 5, of WebSphere Operational Decision Management for z/OS.

© Copyright International Business Machines Corporation 2012. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices . . . vii
Trademarks . . . viii

Preface . . . ix
The team who wrote this book . . . ix
Now you can become a published author, too! . . . x
Comments welcome . . . xi
Stay connected to IBM Redbooks . . . xi

Chapter 1. The case for WebSphere Operational Decision Management V7.5 . . . 1
1.1 What is Operational Decision Management . . . 2
1.2 When to think about Operational Decision Management . . . 3
1.3 Why operational decision management in my z/OS applications . . . 4
1.4 Where can I use Operational Decision Management . . . 5
1.5 Who is involved in Operational Decision Management . . . 6
1.6 Overview of the scenario used in this book . . . 7

Chapter 2. WebSphere Operational Decision Management V7.5 on z/OS overview . . . 9
2.1 How business rule and event externalization enables application modernization . . . 10
2.2 Key concepts to understand Decision Management . . . 10
2.3 WebSphere Operational Decision Management for z/OS overview . . . 11
2.4 Operational concept . . . 13
2.5 Decision Center for z/OS . . . 15
2.5.1 Features . . . 15
2.5.2 Distribution structure . . . 16
2.5.3 Decision Center console . . . 16
2.5.4 Rule Solutions for Office . . . 17
2.5.5 Decision Center for Business Space . . . 17
2.6 Decision Server for z/OS . . . 18
2.6.1 Distribution structure . . . 18
2.6.2 Features . . . 19
2.6.3 Decision Server Rules . . . 20
2.6.4 Decision Server Events . . . 23

Chapter 3. Getting started with business rules . . . 27
3.1 Overview . . . 28
3.1.1 Business scenario . . . 28
3.1.2 Business model . . . 28
3.1.3 Scenario rule model . . . 28
3.1.4 Project structure of a business rule on z/OS . . . 29
3.2 Getting started from a COBOL copybook . . . 29
3.2.1 Scenario overview . . . 29
3.2.2 Creating a rule project . . . 30
3.2.3 Creating COBOL XOM from a COBOL copybook . . . 31
3.2.4 Creating a business object model from the Java XOM . . . 36
3.2.5 Declaring ruleset parameters . . . 39
3.2.6 Adding BOM methods and mapping them to the XOM . . . 41
3.2.7 Creating the ruleflow . . . 48
3.2.8 Authoring rules . . . 51

© Copyright IBM Corp. 2012. All rights reserved.

3.2.9 Preparing the rule execution . . . 53
3.2.10 Building a COBOL application for rule execution . . . 59
3.3 Getting started from an existing rule project . . . 62
3.3.1 Overview . . . 62
3.3.2 Generating a copybook from the BOM . . . 63
3.3.3 Deploy rule artifacts to zRule Execution Server for z/OS . . . 68
3.3.4 Building a COBOL application for rule execution . . . 69

Chapter 4. Decision server events . . . 73
4.1 Scenario overview . . . 74
4.2 Building the event application . . . 75
4.2.1 Event project overview . . . 76
4.2.2 Creating the event project . . . 76
4.2.3 Creating the business objects from a COBOL copybook . . . 77
4.2.4 Creating the event . . . 80
4.2.5 Creating the action . . . 85
4.2.6 Creating the event rule . . . 86
4.2.7 Configuring the technology connectors . . . 87
4.3 Deploying the event application to the event run time . . . 88
4.3.1 Creating the event runtime connection . . . 89
4.3.2 Deploying the event project to the event run time . . . 90
4.4 Emitting events from CICS . . . 92
4.4.1 CICS event support . . . 92
4.4.2 CICS Event Binding Editor . . . 92
4.4.3 Creating the CICS Bundle project . . . 92
4.4.4 Creating the event binding . . . 93
4.4.5 Creating the event specification . . . 93
4.4.6 Creating the capture specification . . . 95
4.4.7 Defining the adapter . . . 98
4.4.8 Deploying the bundle to CICS . . . 99
4.5 Running the scenario . . . 104
4.5.1 Enabling history in the Decision Server Event run time . . . 104
4.5.2 Sample COBOL application to emit the Request event . . . 105
4.5.3 Emitting the event and firing the FollowUp action . . . 106
4.6 Using connectors to receive events from various z/OS sources . . . 108
4.6.1 Connectors running in the WebSphere Application Server . . . 108
4.6.2 Connectors running as a stand-alone batch job . . . 109

Chapter 5. Managing business decisions through the full lifecycle . . . 111
5.1 What is the lifecycle of rule artifacts in decisions . . . 112
5.1.1 Managing artifacts . . . 113
5.1.2 What roles are involved in the decision lifecycle . . . 116
5.2 Sharing decision artifacts between z/OS and distributed . . . 117
5.3 Installation topologies for Decision Center . . . 118

Chapter 6. Decision testing and simulation . . . 121
6.1 Making the right testing and simulation decisions . . . 122
6.1.1 Verifying the business logic implementation by testing . . . 122
6.1.2 Simulation: Running what-if analysis and fine-tuning the decision logic . . . 123
6.2 Testing and simulation architecture for z/OS decision services . . . 124
6.2.1 Test and simulation artifacts . . . 124
6.2.2 Runtime architecture . . . 130
6.2.3 Development and authoring tools . . . 136
6.3 Testing and simulation lifecycle . . . 136

6.4 Running tests and simulations from Rule Designer and Decision Center 137
6.4.1 Scenario: Testing the insurance eligibility project using the Excel scenario suite format from Rule Designer and Decision Center 137
6.4.2 Scenario: Creating a custom scenario suite format to run simulations in Decision Center using input data from a VSAM file. 173

Chapter 7. Advanced topics for decision authoring 201
7.1 Designing the decision interface 202
7.2 Mapping from the COBOL copybook 202
7.2.1 Structure of a COBOL-based rule project 203
7.2.2 Supported COBOL data types 205
7.2.3 Creating custom converters 206
7.2.4 Mapping level-88 constructs into BOM domain types 208
7.3 Starting from an existing Java-based BOM project 210
7.3.1 Mapping Java data structures to COBOL 211
7.4 Extending the capability of the rule execution with BOM methods 213
7.4.1 Preferred practices for using virtual methods 213
7.4.2 Calling out from a ruleset to a VSAM file to augment data 215
7.5 Considerations for sharing rules between z/OS and distributed applications 218
7.5.1 Sharing a COBOL-based project with Java applications 218
7.5.2 Sharing a Java BOM-based project with COBOL applications on z/OS 219
7.6 Coding the COBOL client application 220
7.6.1 HBRWS header structure 220
7.6.2 HBRCONN API call. 222
7.6.3 HBRRULE API call 222
7.6.4 HBRDISC API call. 222
7.7 Authoring considerations for performance 222

Chapter 8. Runtime configuration options for Decision Server on z/OS 223
8.1 Overview 224
8.1.1 Configuring the run times 224
8.1.2 Prerequisite checklist 225
8.2 Running on z/OS stand-alone 226
8.2.1 Standalone zRule Execution Server for z/OS 226
8.2.2 Configuring the Standalone zRule Execution Server for z/OS 227
8.2.3 Creating the data sets that are changed for the zRule Execution Server for z/OS instance 227
8.2.4 Creating the Working Datasets using HBRUUPTI 233
8.2.5 Creating the working directories in UNIX System Services 235
8.3 Configuring the Standalone zRule Execution Server for z/OS (one zRule Execution Server for z/OS/one console). 236
8.3.1 Defining a new subsystem for zRule Execution Server for z/OS 236
8.3.2 Creating the started tasks (HBRXCNSL and HBRXMSTR) 236
8.3.3 Securing the zRule Execution Server for z/OS for z/OS resources 239
8.3.4 Starting the new instance 244
8.3.5 Logging in to the zRule Execution Server for z/OS console and performing diagnostics 245
8.4 A single RES console managing multiple zRule Execution Server for z/OS instances on one LPAR 247
8.4.1 Adding another zRule Execution Server for z/OS to an already running zRule Execution Server for z/OS console. 247
8.4.2 Updating HBRINST to add a zRule Execution Server for z/OS to an existing zRule Execution Server for z/OS console. 247

8.4.3 Creating the working directory 249
8.4.4 DB2 persistence 249
8.4.5 Creating the new subsystem for the new Standalone zRule Execution Server for z/OS 249
8.4.6 Modifying and adding the started tasks to the PROCLIB 249
8.4.7 Security setup for the new Standalone zRule Execution Server for z/OS. 251
8.4.8 Starting the new instance 251
8.5 Configuring the zRule Execution Server for z/OS running on a CICS JVM server . . . 251
8.5.1 Updating HBRINST for the CICS zRule Execution Server for z/OS 251
8.5.2 Creating the working directories 252
8.5.3 Defining a new subsystem for CICS JVM zRule Execution Server for z/OS . . . 253
8.5.4 Modifying and adding the started tasks to the PROCLIB 253
8.5.5 Creating the JVM profile 255
8.5.6 Defining the CICS resources. 255
8.5.7 Adding HBRLIST to the system initialization table 255
8.5.8 Changing the CICS region JCL. 255
8.5.9 Security for the zRule Execution Server for z/OS on CICS JVM server 256
8.5.10 Starting the zRule Execution Server for z/OS on CICS JVM server instance. . 256
8.5.11 CEDA installation of HBRGROUP resources 257
8.5.12 Database connect for the CICS region 257
8.5.13 Connecting the zRule Execution Server for z/OS to the JVM server 257
8.5.14 Deploying the installation verification program 257

Appendix A. Additional material 259
Locating the web material 259
Downloading and extracting the web material 259

Abbreviations and acronyms 261

Related publications 263
IBM Redbooks 263
Other publications 263
Online resources 263
Help from IBM 264

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2012. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	ILOG®	Redbooks®
CICS Explorer®	IMS™	Redbooks (logo) ®
CICS®	Orchestrate®	System z®
DB2®	RACF®	WebSphere®
IBM®	Rational®	z/OS®

The following terms are trademarks of other companies:

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

The IBM® WebSphere® Operational Decision Management product family provides value to organizations that want to improve the responsiveness and precision of automated decisions. This decision management platform on IBM z/OS® provides comprehensive automation and governance of operational decisions that are made within mainframe applications. These decisions can be shared with other cross-platform applications, providing true enterprise decision management.

This IBM Redbooks® publication makes the case for using WebSphere Operational Decision Management for z/OS and provides an overview of its components. It is aimed at IT architects, enterprise architects, and development managers looking to build rule-based and business event-based solutions. We provide step-by-step guidance on getting started with business rules, and in creating business events, taking a scenario-based approach. This book provides detailed guidelines for testing and simulation, and it describes advanced options for decision authoring. Finally, we describe and document multiple runtime configuration options.

The team who wrote this book

This book was produced by a team of specialists from around the world.

Janet Wall is the Product Line Manager for both the IBM WebSphere Operational Decision Management products (combined execution and management of Business Events and Business Rules) and IBM Business Process Management (BPM) focused on IBM System z® solutions. She has 35 years experience in IBM CICS®, IBM IMS™, IBM DB2®, and COBOL development and z/OS system programming. She has an MBA in Finance and a BS in Computer Science from Seton Hall University. Janet was a contributing author of *Business Rules Applied*, by Barbara Hale. Janet has authored several Business Rules and Business Rule Mining articles. She has presented Business Rule Management and Information Architecture at several IT conferences. Janet has authored several white papers about Business Rules on z/OS for incremental modernization. She has also co-developed the best practices for Business Rule Mining from COBOL code to design into a Business Rules Management System (BRMS).

Chris Backhouse is a Senior Software Engineer in the Hursley development laboratory. He joined IBM 12 years ago after gaining a degree in Computer Engineering from the University of Southampton. Chris was part of the CICS team where he worked on Business Events, Web services, and the Service Flow Runtime. Chris currently has architectural responsibility for business rules and events on System z as part of the WebSphere Operational Decision Management for z/OS product. He started working in this role after IBM acquired ILOG® three years ago.

Sebastien Brunot is a Software Architect, Software Developer, and Test Specialist in the UK. He has 11 years of experience in the Java and testing fields. He holds an MS in Computing and Aeronautics from the National School of Civil Aviation. His areas of expertise include Java software design and development, unit testing, functional testing, performance testing, and system verification testing.

Pierre Feillet is an Architect in France for IBM WebSphere Operational Decision Management products. He has 15 years of experience in the Business Rules field in IBM and

© Copyright IBM Corp. 2012. All rights reserved.

ILOG. He holds a degree in Computer Sciences. His areas of expertise include Decision Management and high-speed decision run time in distributed and z environments.

Mark Hiscock is the WebSphere Operational Decision Management technical lead for z/OS based in Hursley, UK. He has worked at IBM for 11 years and holds a degree in Computer Science from the University of Portsmouth. His expertise includes z/OS, Decision Management, Messaging Middleware, and Message Brokering.

Mike Johnson works as a Developer in the WebSphere Operational Decision Management for z/OS System Test and Development team. He is based in the IBM Hursley development laboratory in the UK.

Yongxin Pan is a Staff Software Engineer in the IBM China Development Lab (Shanghai). He has nine years of experience in Java development. He holds a Masters degree in Mechanical Engineering from Zhejiang University, China. His areas of expertise include BRMS and WebSphere Operational Decision Management on z/OS.

Wang Wenjie is an Advisory Software Engineer in the IBM China Development Lab (Shanghai). He has over 10 years of experience in Java development and BRMS. His areas of expertise include WebSphere Operational Decision Management on z/OS, Java enterprise application development, and service-oriented architecture (SOA).

Timothy Wuthenow is an IT Specialist for WebSphere on System z. He holds a degree in Chemical Engineering from North Carolina State University. His area of expertise is in the configuration of WebSphere Operational Decision Management on z/OS and CICS. He has developed customer demonstrations using the new capabilities of WebSphere Operational Decision Management within the CICS and z/OS environments. He was a contributor to the *Introduction to the CICS Ecosystem*, SG24-7952, Redbooks publication.

This project was led by:

- Martin Keen, IBM Redbooks Project Leader

Thanks to the following people for their contributions to this project:

- Kieron Brear, WebSphere Operational Decision Management on z/OS Development Manager
- Steve Demuth, WebSphere Decision Management Architect
- Jackie Scott, CICS TS Development
- Lewis Evans, WebSphere Business Events Development
- Dennis Weiand, IBM Dallas Systems Center
- Brett Stineman, IBM WebSphere Product Marketing

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

The case for WebSphere Operational Decision Management V7.5

This chapter introduces operational decision management. We describe using WebSphere Operational Decision Management V7.5 for z/OS to address the agility needs of an organization's CICS and batch COBOL applications. We describe the following topics:

- What is Operational Decision Management
- When to think about Operational Decision Management
- Why operational decision management in my z/OS applications
- Where can I use Operational Decision Management
- Who is involved in Operational Decision Management
- Overview of the scenario used in this book

© Copyright IBM Corp. 2012. All rights reserved.

1.1 What is Operational Decision Management

Smarter business outcomes require the ability to quickly adapt to change. But, corporate leadership in every industry is struggling to keep pace with change especially in this less predictable, complex economic environment. These changes directly affect the corporation's business policies and those business decisions that are required to consistently apply business policies. A *business policy* is a statement of guidelines governing business decisions. For example, a bank might have a lending policy stating, "Customers whose credit rating is above average are entitled to a discounted rate on their loan". The traditional application development lifecycle requires a business analyst to document the detailed requirements to design and develop this policy into one or more business applications. Then, one or more developers take those requirements and code or embed the requirements into the various application programs. The development is then followed by a lengthy testing process. Unfortunately, the decisions are now hidden in the code in one or more programs, and over time as additional changes are added to the business policy, the code becomes more complex, making it difficult for change and traceability.

Decision management is emerging as an important capability for delivering agile business solutions. Decision Management is the "business discipline, supported by software that enables organizations to automate, optimize and govern repeatable business decisions improving the value of customer, partner and internal interactions." Decision Management is the tool to help corporations accelerate their reaction to the pace of the growing complexity of business changes.

Accurate real-time business decisions provide many benefits within an organization. Better decisions help companies identify opportunities for increased revenue and profitability, such as in marketing and sales. Better decisions also help companies enforce compliance with external and internal policies, such as in claims processing or eligibility determination. Finally, better decisions help companies manage and reduce risk, such as with fraud detection and credit approvals.

Figure 1-1 on page 3 shows the two emerging forms of decision management technology as described in *Making Better Decisions using WebSphere Operational Decision Management*, REDP-4836. The objective of this book is to describe how Operational Decision Management capabilities (a combination of business rules and business events shown in Figure 1-1 on page 3 boxed in red) can address the agility needs of organization's CICS and batch COBOL applications.

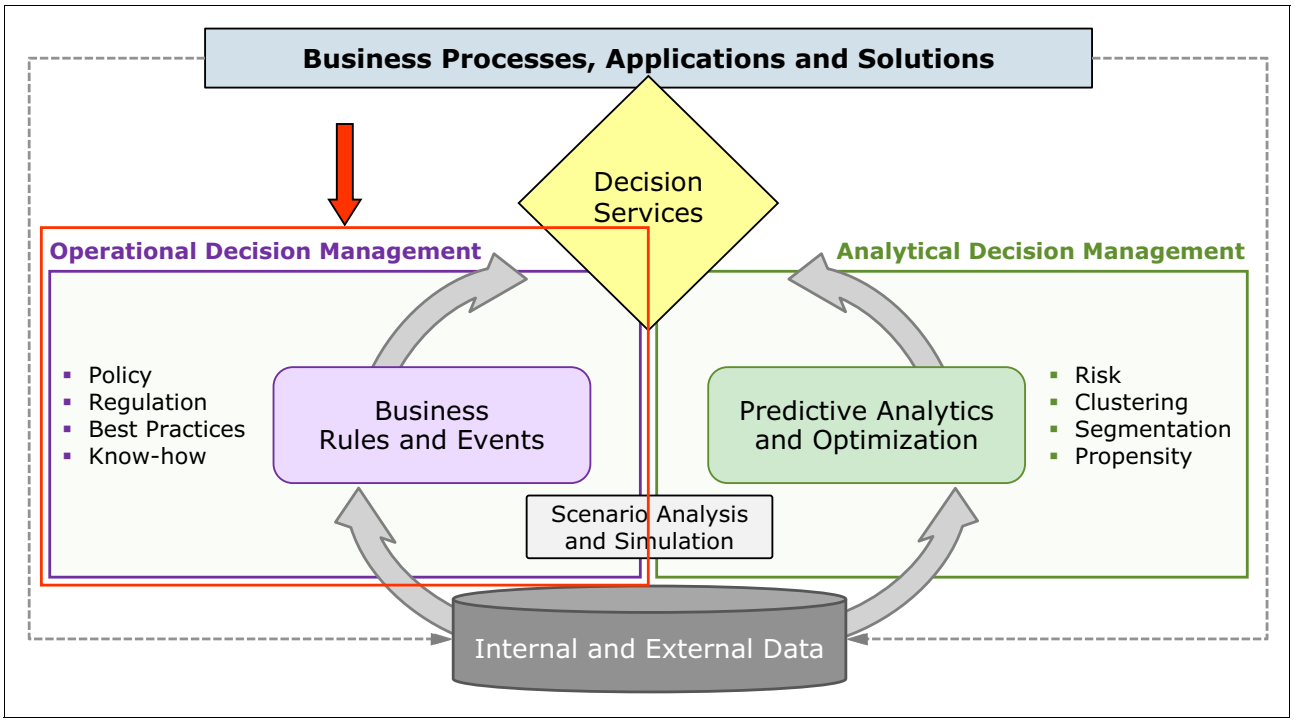


Figure 1-1 What is Decision Management

1.2 When to think about Operational Decision Management

Organizations today have their core business processes automated in application systems that were developed over the course of years or even decades, making the applications difficult for a person to comprehend. As these software assets mature, they tend to become increasingly complex. Unfortunately, this complexity is compounded by a decline in technical and business understanding of how these assets support business goals and priorities.

Corporations need to identify solutions that support business change and cycles far more effectively than traditional application development methods and take direct advantage of business expertise. A decision management approach separates decision logic from application code so that the business logic can be modified without affecting the rest of the application. This approach enables users to assess, implement, test, and deploy changes quickly, allowing them to react to market conditions or new regulations in a timely manner.

Business decisions are made every day in business transactions incorporated in web or online business applications and batch applications. These business decisions can be categorized in three categories:

- The first type is identifying opportunities to increase revenue and profitability. This type includes decisions that are used by marketing and sales to make targeted offers based on customer profiles, demographics, and analytical models.
- The second type of automated decision concerns consistency and compliance. This type of automated decision can be found in all industries, such as financial, insurance, and government sectors. Examples are claims validation and *straight-through processing* (the ability to process a claim without manual intervention). A key goal for many organizations is not only to automate the claims process but generally to improve the quality of claims validation, while simultaneously speeding up response time, saving money, and improving

customer satisfaction. The same goal is true for compliance-related decisions, such as eligibility determination or payment processing.

- The third type of automated decision includes those decisions that help reduce and mitigate risk. Examples are credit decisioning, fraud detection, and insurance underwriting.

1.3 Why operational decision management in my z/OS applications

Organizations embark on application modernization projects to focus on how their core System z business applications can respond rapidly to emerging opportunities. To manage agile solution delivery, it is essential to be able to understand these business applications in terms of the business decisions they implement and the effect of decision changes on key business processes.

Organizations can rapidly and efficiently advance into their application modernization projects by incrementally externalizing their business decisions from COBOL applications and moving them into a decision management system. Most companies begin using Operational Decision Management with one or possibly two business decisions at a time. Examples are deciding when to reorder products in a specific region and identifying the eligibility of a new customer. Taking an incremental approach with decision management in your core business applications provides organizations with a return on investment (ROI) in their first phase of their projects. An incremental approach avoids embarking on a lengthy, labor-intensive “rip and replace” project. An incremental approach also enables the team to understand the design and management techniques of decision management.

Operational Decision Management combines the authoring, testing, and management of business rules and business events that are required for implementing business decisions. Operational Decision Management enables organizations to adapt incrementally the business decisions in their System z while avoiding lengthy application development cycles.

Operational Decision Management offers these features:

- A set of tools for business users, administrators, and developers to edit and manage rules
- A powerful Decision engine to execute business decisions
- A robust Decision repository to tie everything together
- An extensive library to define and extend the decision execution and management environment

Applying Operational Decision Management to application modernization projects can incrementally address projects in the following areas:

- Effective application maintenance: z/OS development teams need to address their long list of maintenance projects for their core COBOL business applications. If a maintenance project requires updates to the decisions that are implemented in a specific application (for example, calculating preferred customer discounts or determining credit fraud), redesign those rules in Operational Decision Management for enhanced ongoing management.
- Consolidating or restructuring existing applications: Most organizations have duplicate functionality in multiple applications, which is a high cost whenever changes occur. Duplicated functionality causes a company to spend more time and resources maintaining applications than is necessary. Consolidation combines the same functionality into a core business application. Incorporating Operational Decision Management technology into

those modernization projects centralizes the business decisions that were redundant in the duplicated function into one source for ease of change and management.

- Sharing business decisions across applications and platforms. This area is an effective way to obtain a higher ROI. WebSphere Operational Decision Management for z/OS provides the tooling to design business decisions for your COBOL applications that can be reused or shared for execution in Java applications on z/OS or distributed platforms.

1.4 Where can I use Operational Decision Management

Operational Decision Management combines the management of business events and business rules and, in this combination, enables intelligent and responsive decision automation. It enables organizations to flexibly build solutions that can detect and react to event patterns as they occur within a specified time period. Then, it provides the appropriate automated decision response to transactional and process-oriented business systems.

Business Event Processing focuses on sense and response. Business Event Processing executes a continuous evaluation of events from multiple sources. It attempts to detect patterns of events that occur or do not occur as expected that represent situations to which the business wants to respond. Business Event Processing is primarily focused on situational awareness by detecting patterns in the overall flow of events in the enterprise.

Business rules are the specific statements that enforce a policy. The policies are translated into business rules, which are the detailed conditions and actions that unambiguously enforce the policy. The business rules expand upon the policy by stating in detail the circumstances under which the policy is applicable and the actions that enforce it. Business decisions can be defined that result in new application behaviors, offering a quick and productive route to enhance the business responsiveness of CICS and IMS operations. Also, by putting the business owner in control of the management of business decisions and at the same time retaining good governance and change management.

Designing, maintaining, testing, implementing, and managing business decisions within Operational Decision Management provides these benefits:

- A convenient communication channel between IT and business teams
- Easy implementation and reuse of business decisions across the enterprise
- Flexible options for progressive IT modernization

Operational Decision Management can benefit your organization in many ways:

- Customer relationship: By improving customer interaction and personalization:
  - Achieve finer-grained personalization in customer interactions. Business rules enable business users to implement more tailored promotions, pricing, risk models, and so on, therefore, increasing the precision and personalization of operational decisions
  - Move decision-making to the point of contact with customers and enable enterprises to deploy decisions at the contact point with customers and improve consistency in decisions about customers and customer interaction
- Enterprise processes: By improving business alignment, compliance, and transparency:
  - Achieve high pass-through rates in process automation. Centrally managing business decisions enables you to streamline processes and helps you achieve higher levels of automation and higher pass-through rates by externalizing decisions and automating more complex decisions
  - Maximize decisions for resources, risk, and value. Managed business decisions enable businesses to tie sources of insight (from historical data, predictive knowledge,

simulation, and events) and decision automation capabilities to achieve consistently better business outcomes and maximize resources and value

- Business agility and speed: By improving business-led agility and responsiveness:
  - Empower business users to manage and improve decisions. Managed business decisions provide an agile platform to enable business users to manage decisions and changes in a short time frame
  - Shorten response time to changing market conditions and events
  - Increase enterprise responsiveness to unforeseen events, as well as shortened response time and time-to-business due to higher levels of automation

1.5 Who is involved in Operational Decision Management

For a team to be effective, it is necessary to have the right set of skills on the team or available to the team for consultation. As stated in *Making Better Decisions using WebSphere Operational Decision Management*, REDP-4836, the responsibility for specifying and managing the business needs to be considered from the point of view of the business roles.

Business analysts are responsible for specifying how the business needs to behave, identifying key performance indicators that reflect how well the business is doing, and defining the processes and decision points needed to manage the business.

Line of business users are responsible for the day-to-day management of the business using the solutions. They are responsible for monitoring the key performance indicators (KPIs) and modifying the way decisions are made in order to optimize the business. In a Decision Management solution, these roles have the responsibility for optimizing decisions to meet the business need.

Users are responsible for using the solution and need to consider the solution from a consumability and process-efficiency perspective. In many cases, the user role might be the subject of KPIs that the solution is designed to support.

The responsibility for the delivery and maintenance of these systems lies with the IT department. When embarking on a project using Operational Decision Management technology, there are new and expanded project roles. We list several key roles to include in these projects:

- Application subject matter expert (SME)

  An SME on the application that is being mined is an essential member of the team. This individual provides awareness of the programming styles in use and an understanding of the role that the application serves. Ideally, this SME is aware of the application's programming history, including the original purpose and design and how the application changed over time.

- Enterprise architect

  The enterprise architect can provide valuable context for the application's role in the Business Decisions that are managed.

- Business rule analyst/rule designer

  This role understands the business decisions and rules for the new implementation. This person provides input to the new design of the rules. This role is normally combined with the business rule designer/implementer.

- Business object modeler

This role defines the business object model for the target application and maps the COBOL structures to business friendly vocabulary.

- Business rule repository administrator

  This role is responsible for ensuring that the business object model and rules are defined consistently for all phases of the project and ensures that the rules can be shared across platforms.

- Business rule miner

  This role is optional. Normally, this person is the COBOL developer or a technical business analyst responsible to “drive through the code” to identify the candidate business rules. This person enters them in the rule authoring user interface.

1.6 Overview of the scenario used in this book

The scenario that illustrates how Operational Decision Management can be implemented in an existing core CICS and batch COBOL business application is based on a fictional insurance company. The insurance company needs to provide greater agility to sense, respond to, and decide when and what to do, when a customer needs to contact the insurance company. In addition, the company currently is unable to detect when the same customer or household contacts it from multiple channels of the company. This issue caused customer dissatisfaction, as well as many fraud situations of multiple requests for quotes from the same customer or household.

There are disparate business applications that manage customer channels. The call center uses a web application, and the branch offices use a CICS application to manage the insurance company’s claims business processes. The insurance company initiated an application modernization project to focus on its business decisions that relate to customer contact within these channel applications.

The use of business rules ensures more consistent results and can detect fraud when a customer household submits similar quote requests. Business rules for customer validation and fraud detection need to be shared across the applications. Business rules can be used for customer profiling: identify the customer, its products with the company, and the previous quotes provided to the customer on certain products.

Business Events in the CICS application and web applications can be used to identify if there are similar requests for quotes and alert the insurance officer of possible fraud. The applications can reject a request for a quote if the number of events in a specific amount of time on the same product is reached.

The following chapters use this scenario. We describe how to use WebSphere Operational Decision Management for z/OS for both the business rules and business events for the web and CICS COBOL applications to address the insurance company application modernization project. See Figure 1-2 on page 8.

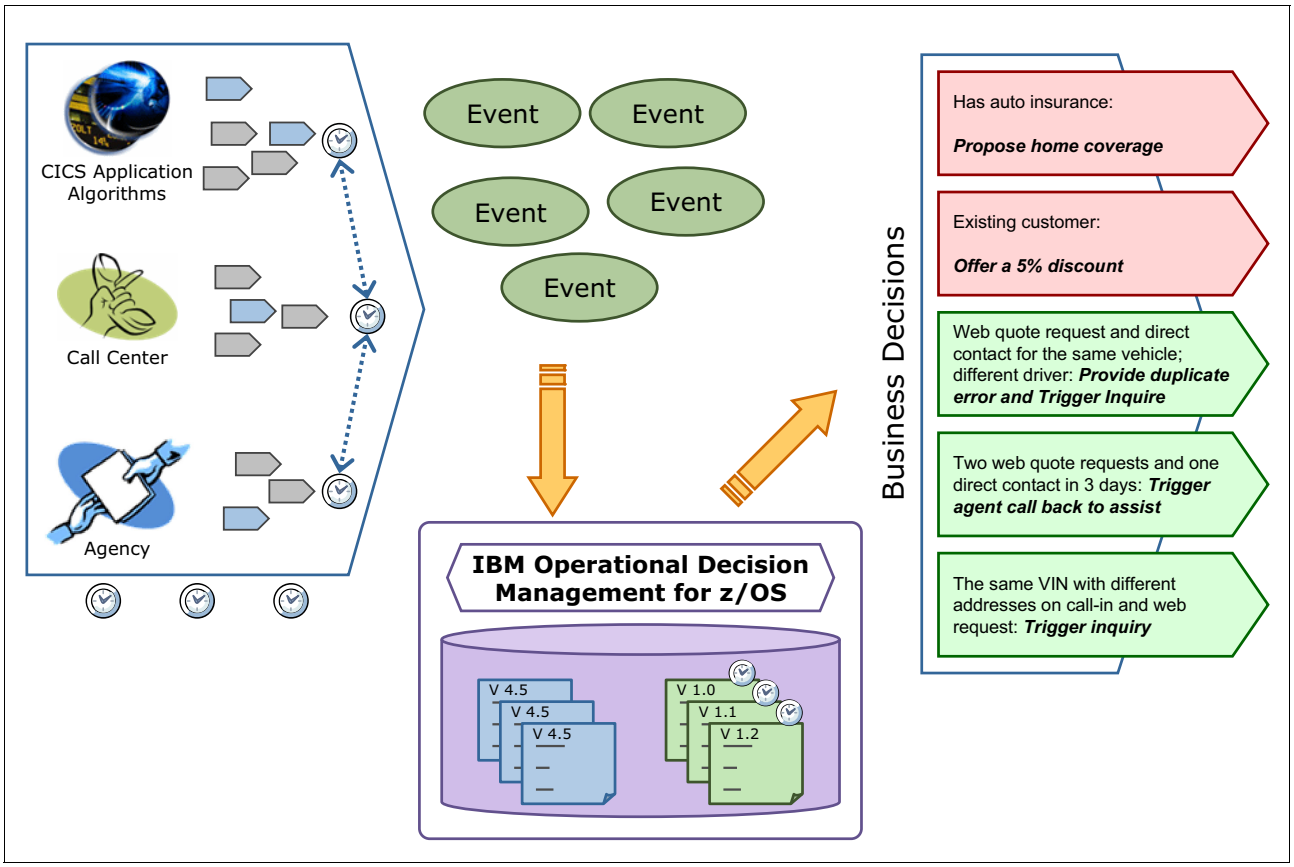


Figure 1-2 Insurance company application modernization project

2

WebSphere Operational Decision Management V7.5 on z/OS overview

This chapter provides an overview of IBM WebSphere Operational Decision Management V7.5 on z/OS. It contains the following sections:

- How business rule and event externalization enables application modernization
- Key concepts to understand Decision Management
- WebSphere Operational Decision Management for z/OS overview
- Operational concept
- Decision Center for z/OS
- Decision Server for z/OS

© Copyright IBM Corp. 2012. All rights reserved.

2.1 How business rule and event externalization enables application modernization

Companies can more rapidly and efficiently advance their application modernization projects by incrementally externalizing business rules and events from COBOL applications and moving them into a decision management system. WebSphere Operational Decision Management for z/OS can play a major role in application modernization initiatives:

- Refactor/application structure modernization. Business rules and events are externalized from core application code into a layered and modular architecture for better maintainability, reuse, and service enablement. Business rules can then be executed through a call to the appropriate Decision Server run time or through the execution of a native COBOL procedure by a COBOL-generated subprogram.
- Documentation modernization. After the business rules and events are externalized and managed using WebSphere Decision Center, it continues to manage and document all the changes through its version control and audit management features. The natural-language format of the rule authoring allows the rule implementation to be the documentation. Through the Decision Center's Rule Solution for Office component, business rules and their associated metadata can also be integrated into Microsoft Word and Excel documents.
- Reuse modernization. Business rule and event externalization helps identify the reusable components of the existing application and facilitates the reuse of business decisions that span applications. Extracting business rules from application code empowers business user subject matter experts (SMEs) to manage the rules in a manner that focuses on how decision logic is used to support overall business objectives, regardless of individual applications. This method enables a decision logic change to be implemented one time and then easily deployed by any application that requires it. WebSphere Operational Decision Management for z/OS is designed to increase business agility by helping to revitalize your organization's existing application portfolios. By providing multiple rule and event execution capabilities, it offers flexible options when adopting a business rules approach for application modernization on mainframe systems.

2.2 Key concepts to understand Decision Management

IBM WebSphere Operational Decision Management V7.5 for z/OS provides comprehensive automation and governance of operational decisions that are made within mainframe applications.

To explain how WebSphere Operational Decision Management V7.5 for z/OS works, we explain its nomenclature. Every organization has people responsible for setting the policies by which they do business. In this context, a *business policy* is a statement of guidelines governing business decisions. An insurer might have an underwriting policy, for example, that says "an underage applicant for insurance on high-powered sports cars is ineligible for coverage."

A general policy statement is not sufficient to form the basis of an automated decision. A policy manager must translate the policy into more specific statements that specify the details of how the policy is enforced. In this insurance example, the policy manager is a SME for the automobile underwriting domain.

The specific statements that enforce the policy are *business rules*. Business rules are translations of the policies into detailed conditions and actions that unambiguously enforce decision outputs. We must start with an understanding of the data, interactions, and terminology included in the specific domain of the business policy. The understanding of this information leads to the definition of a vocabulary that is used to write the rules. These rules are used by various business systems that require them through an object model, which is developed by IT and mapped to specific data sources within the software infrastructure. To the policy manager, the interaction of the rules with the object model is through the non-technical vocabulary, allowing the policy manager to author and maintain rules using a business syntax. Business rules expand on policies, by stating the detailed circumstances under which the rule is applicable and the actions that enforce it. One policy can translate into many business rules.

Another concept relating to policies is *business events*, which focus on occurrences of significance in a process or transaction that result in a change of state. Examples are, in the auto insurance scenario, the submission of an application into the underwriting system, or the change of an application from “pending” to “accepted”. Definitions describing patterns of interest across multiple business events can be defined to enforce business policies or other organizational objectives. These definitions also take the form of conditions and actions, although the conditions tend to be based on the aggregation of multiple occurrences or correlating event patterns that span a period of time. Business rules, however, tend to be based on an individual occurrence at a specific point in time.

Because both the definition of business rules and business event patterns use a common condition-action form, we refer to both of them as rules. We use the terms “business rules” and “event rules” to distinguish between them.

Even one policy domain, such as personal auto insurance underwriting in our example, can require hundreds or thousands of rules. The rules frequently change over time and differ throughout jurisdictions, customers, products, channels, or any other partition of a business policy domain. The process by which a company manages and governs changes to policies is called the *rule lifecycle*.

2.3 WebSphere Operational Decision Management for z/OS overview

The IBM WebSphere Operational Decision Management product family provides value to organizations that want to improve the responsiveness and precision of automated decisions on z/OS and distributed applications. On z/OS, this decision management platform provides comprehensive automation and governance of the operational decisions that are made within mainframe applications.

WebSphere Operational Decision Management V7.5 for z/OS consists of two orderable products, illustrated in Figure 2-1 on page 13, which together form a platform for the management and execution of business rules and event rules:

- IBM WebSphere Decision Center for z/OS provides an integrated repository and management components for line-of-business (LOB) SMEs to directly participate in the governance of business rule-based and business event-based decision logic. Through the capabilities of WebSphere Decision Center, business and IT functions can work collaboratively. They align the entire organization in the implementation of automated decisions and accelerate the maintenance lifecycle as they evolve, based on new external and internal requirements.

  Decision Center covers these features:

  - Comprehensive decision governance, including role-based security, custom metadata, multiple branch release management, non-technical testing and simulation, and historical reporting
  - Team collaboration through multiple user access for business users and integrated synchronization between IT and business user environments

  Decision Center packaging includes these environments and tools:

  - Decision Center console
  - Decision Center repository
  - Decision Center for Business Space
  - Rule Solution for Office

- IBM WebSphere Decision Server for z/OS provides the runtime components to automate event and rule-based decision logic on mainframe systems. This product enables the detection of actionable business situations and the response of precise decisions based on the context of each interaction.

  With WebSphere Decision Server for z/OS, an organization can monitor a business network to discover and act on event-based data patterns. Then, an organization can process this information against hundreds or even thousands of business rules to determine how to respond within both front-end and back-end systems.

  This product includes these components:

  - Specific run times

    These run times are designed to handle the unique aspects of business rule and business event execution. For business rule execution, this product offers several mainframe runtime options. These options allow development teams to choose a deployment strategy that best fits their mainframe applications and architecture.

  - Eclipse-based development tooling

    Rule Designer and Event Designer provide application development environments, sharing a similar high-level approach and technology.

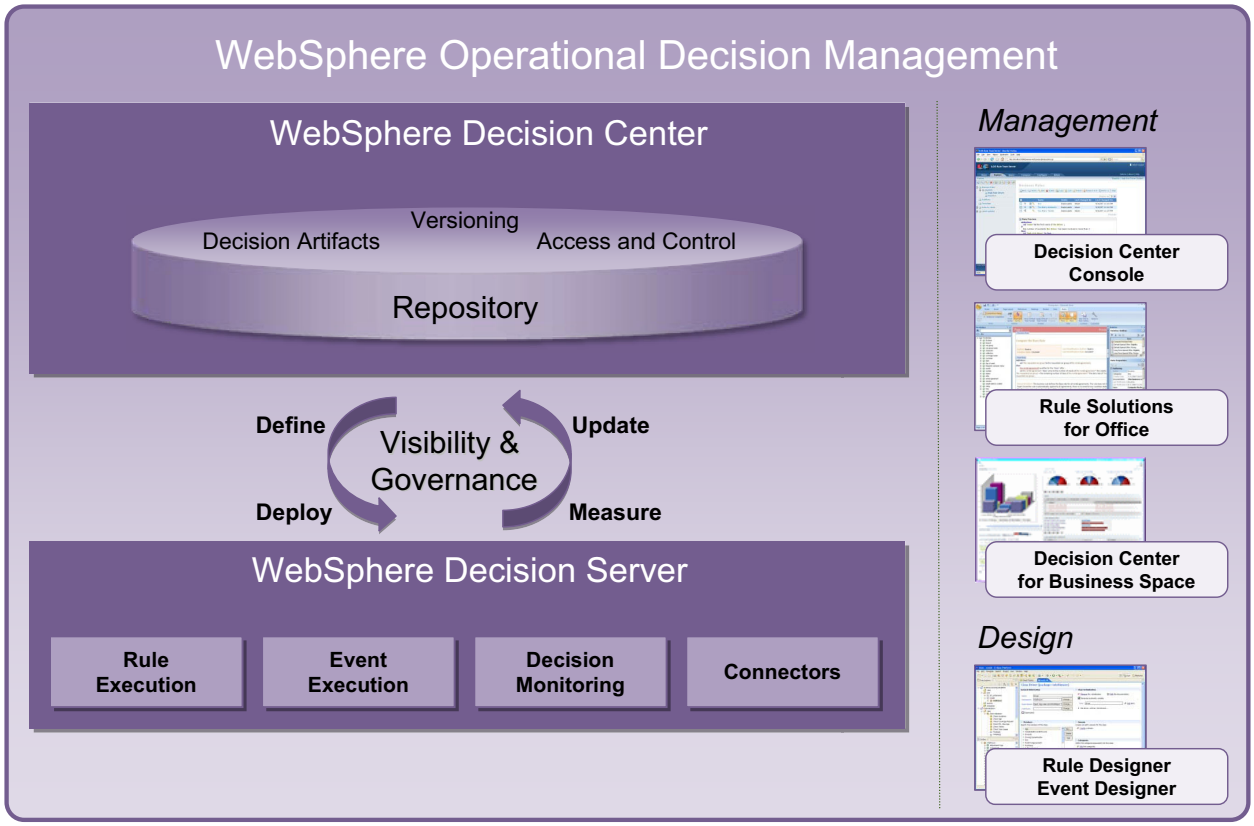


Figure 2-1 WebSphere Operational Decision Management V7.5 for z/OS

2.4 Operational concept

WebSphere Operational Decision Management V7.5 for z/OS delivers the decision management features and applies them on mainframes.

WebSphere Operational Decision Management V7.5 for z/OS includes these products and modules:

- Rule Designer is used as the starting point to create the model on which you author the business rules. Rule Designer is the Eclipse-based development toolkit for business rules. It is installed on a workstation.
- Event Designer is used as an entry point to develop event rules. It consists of the Eclipse based-development toolkit for event rules and is installed on a workstation.
- Decision Center is used as the team repository to govern the business rules and event rules, and to author them through a web interface. Decision Center runs on WebSphere Application Server on z/OS, Linux for System z, or a distributed environment.

  Decision Server run times are split into two types:

  – For Business rules, three approaches are possible: zRule Execution Server for z/OS, Rule Execution Server running on WebSphere Application Server on z/OS, or COBOL source generation. The core runtime stack is common across Rule Execution Server and zRule Execution Server for z/OS, which share the rule engine. zRule Execution Server for z/OS is a runtime solution to manage and operate decision services that are invoked from COBOL applications running in batch and CICS on z/OS.

– For Event rules, a run time exists on WebSphere Application Server on z/OS to process business events.

Figure 2-2 shows the relationships among the various modules and the separation between development and execution.

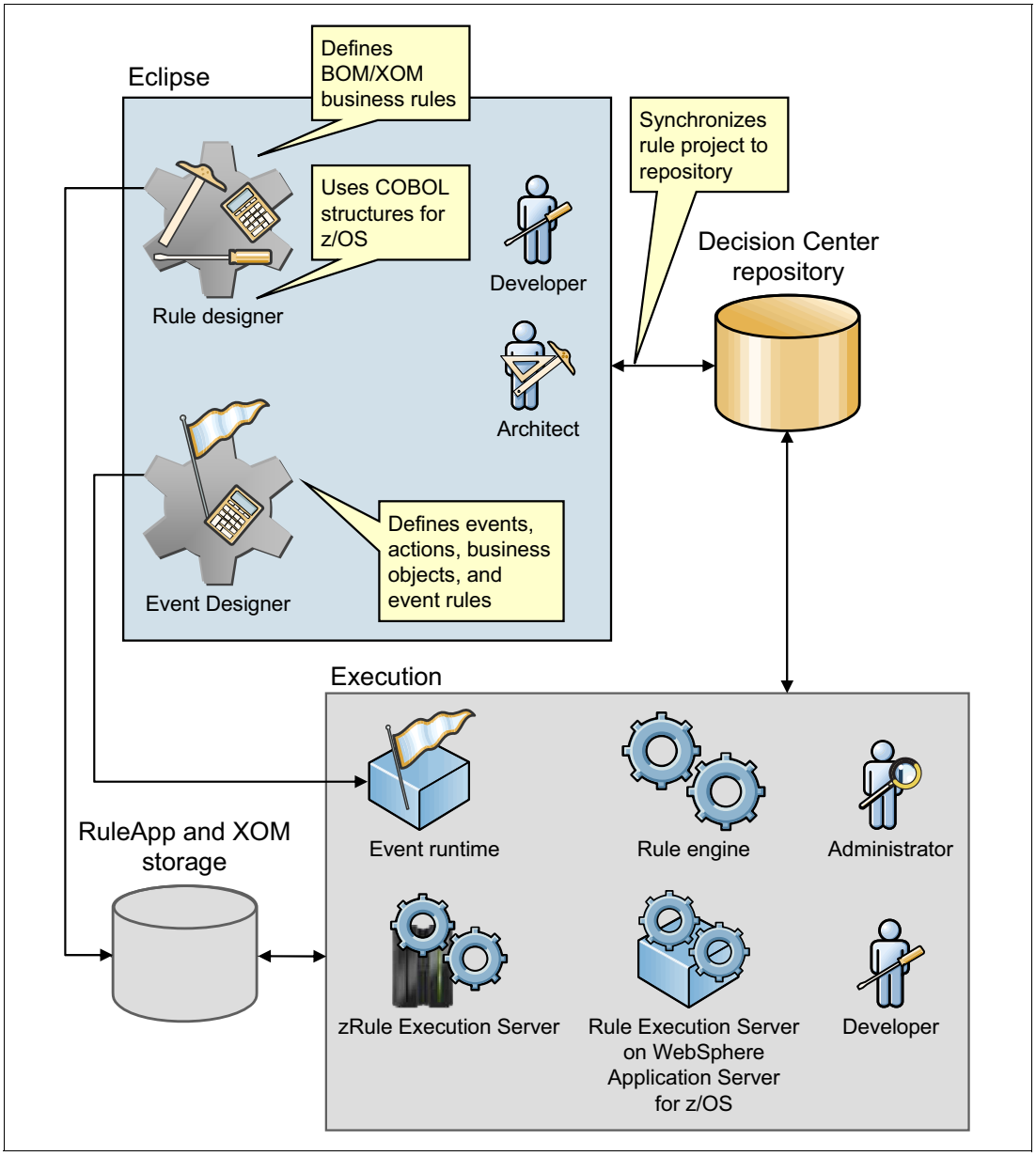


Figure 2-2 Operational concept

Table 2-1 on page 15 shows the tasks across WebSphere Operational Decision Management tools and environments.

Table 2-1 Tasks across WebSphere Operational Decision Management tools and environments

Tasks	Business rules	Event rules
Synchronizing	Rule Designer	Event Designer
Authoring	Decision Center console Decision widget Rules Solution for Office	Decision Center console Decision widget
Reviewing and managing	Decision Center console Decision widget	Decision Center console Decision widget
Validating	Decision Center console	Testing widgets: Event Tester Event Capture Event Replay Monitoring widgets: Event Chart Event Chart Manager Event Layout
Deploying	Decision Center console Decision widget	Decision Center console Decision widget
Administering	Decision Center repository Decision Center console	Decision Center repository Decision Center console

2.5 Decision Center for z/OS

Decision Center provides an integrated repository and management components for LOB SMEs to directly participate in the governance of decision logic. Through the capabilities of WebSphere Decision Center, business and IT functions can work collaboratively. Decision Center helps you to align the entire organization in the implementation of automated decisions. It helps you to accelerate the maintenance lifecycle as the automated decisions evolve based on new external and internal requirements.

2.5.1 Features

Decision Center is the central hub that coordinates the decision lifecycle across the business and IT parts of your organization. It provides the following features for business users to manage their decisions:

- Rule authoring

  Decision Center includes editors for both business rules and event rules. These editors are available in a web console and in Business Space. They are also available in Microsoft Office for business rules only.

- Rule synchronization between users and developers

  Synchronization is the key to collaborative work between business and IT users. You can adopt a developer-centric or a business user-centric approach to managing synchronization.

- Rule review and management

  In the Decision Center console and in the Decision widget, business users can run queries and publish reports on the content of their projects. Decision Center provides ways to customize how business users can view the items in their projects with smart folders.

Business users can also manage releases and work in progress with branches and baselines.

- Rule validation

  Decision Center provides tools for validating that decisions are implemented as expected. For business rules, Decision Center provides testing and simulation of rulesets. For event rules, the Testing widgets and Monitoring widgets help business users analyze the processing of business events.

- Rule deployment

  Following verification, you can deploy your decision logic as business rules or event rules to the production system.

- Administration

  After you configure Decision Center, you perform several regular administrative tasks to provide optimum service to the business users.

Decision Center can reside on z/OS or Linux for System z. It can be deployed, as well, in a distributed environment to edit the business rules and event rules. It can be connected to a Decision Server running on z/OS.

2.5.2 Distribution structure

The following top-level directories are created under the *<zDC_V7R5M0_InstallDir>* directory for each component in Decision Center for z/OS. These components correspond to modules or other services that provide related functionality or data:

- `events`: This directory contains the necessary resources to configure business space:
  - `bspace`
  - `config`
  - `widgets`
- `teamserver`: This directory contains the necessary resources to deploy and configure Decision Center on the WebSphere Application Server on z/OS:
  - `applicationservers`
  - `WebSphere7`
  - `bin`
  - `lib`
- `shared`: This directory contains the shared third-party tools.

2.5.3 Decision Center console

Decision Center console is in a web application that is shared across business rules and business events. This web application empowers business users to author and manage their business rules and events rules. All decision authoring assets are versioned and persisted in an underlying DB repository (Figure 2-3 on page 17).

Authoring artifacts: Note that the authoring artifacts are transformed into executable assets, and then deployed to a runtime repository. The console can be deployed on z/OS and run on WebSphere Application Server on z/OS. Its deployment is also allowed on Linux for System z and distributed environments.

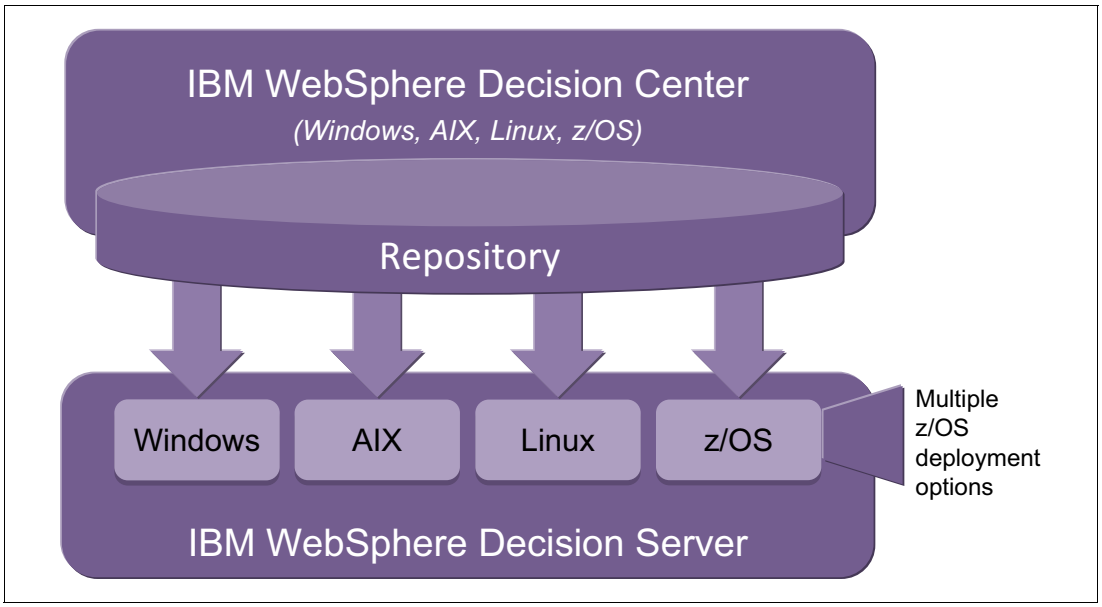


Figure 2-3 WebSphere Operational Decision Management on z/OS environment options

2.5.4 Rule Solutions for Office

Rule Solutions for Office is used for sharing business rules offline. Policy managers and rule authors can author business rules in Rule Solutions for Office. With Rule Solutions for Office, rule authors write rules in Microsoft Word and edit decision tables in Microsoft Excel. They can create mixed rule and non-rule content in a RuleDoc, which retains semantic information together with the actual implementation content of the rules.

You can integrate business rule authoring and management extensions that are developed in Rule Designer into Rule Solutions for Office. The RuleDocs can be synchronized with the Decision Center console.

2.5.5 Decision Center for Business Space

Decision Center comes with a set of widgets to help the authoring and development of the decision logic in a web portal.

The Decision widget is the Decision Center console interface that is available from Business Space. You can integrate rule model extensions that are developed in Event Designer into the Decision Center console and Decision widget.

You can use Business Space widgets that are supplied by Decision Center to monitor your business event processing by defining and viewing charts. To validate the business logic, policy managers can use the Testing widgets. In the Event Tester widget, they can test the event logic in a business process. In the Event Capture widget and the Event Replay widget, they can capture events from a production system and replay a sequence of one or more of them, typically on a test system.

2.6 Decision Server for z/OS

Decision Server for z/OS provides the runtime components to automate decision logic. These components enable the detection of actionable business situations and the response of precise decisions based on the specific context of an interaction. With Decision Server, organizations can monitor a business network to discover and act on event-based data patterns. Then, organizations process this information against business rules to determine how to respond within both front-end and back-end systems.

Decision Server is split into the following independent parts, as illustrated in Figure 2-4:

- Decision Server Rules:
  - Rule Designer, which is an Eclipse-based integrated development environment (IDE) for business rule development
  - Rule Execution Server on WebSphere Application Server on z/OS
  - zRule Execution Server
- Decision Server Events:
  - Event Designer, which is an Eclipse-based IDE for business event development
  - Event Execution Runtime, which runs on WebSphere Application Server on z/OS

Decision Server Rules and Events have their dedicated stacks and executable artifact repositories.

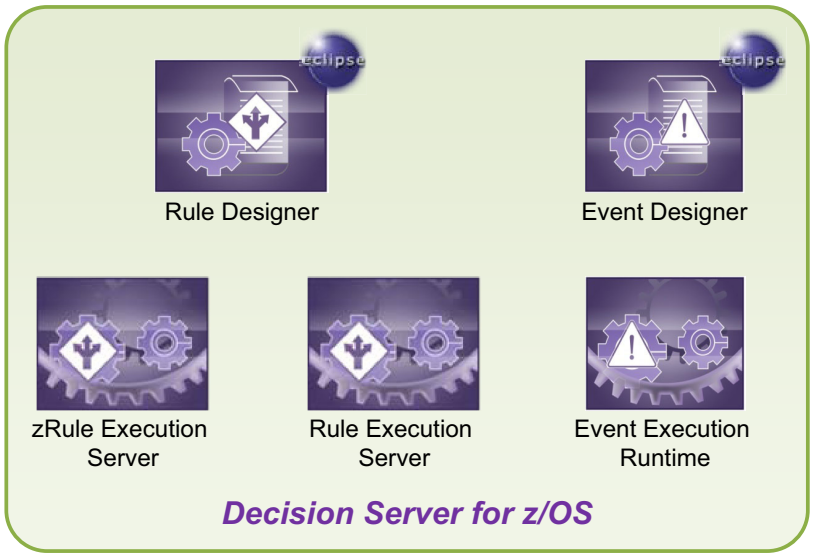


Figure 2-4 Decision Server packaging

2.6.1 Distribution structure

The following top-level directories are created under the *<zDS_V7R5M0_InstallDir>* directory for each component in Decision Server for z/OS. These components correspond to modules or other services that provide related functionality or data:

- `events`: This directory contains the necessary resources to deploy and configure the event run time:
  - `bspace`
  - `config`

  - `connectors`
  - `director`
  - `integration`
  - `license`
  - `widgets`
- `executionserver`: This directory contains the necessary resources to deploy and configure Rule Execution Server on the WebSphere Application Server on z/OS:
  - `applicationservers`:
    - `WebSphere7`
  - `bin`
  - `lib`
- `shared`
- `zexecutionserver`: This directory contains the necessary libraries and files to configure a zRule Execution Server for a z/OS instance.

The following set of Decision Server for z/OS data sets is added to your system during installation:

- HBRHLQ.SHBRAUTH: Authorized program facility (APF)-authorized modules
- HBRHLQ.SHBRCICS: CICS modules
- HBRHLQ.SHBRCOBC: COBOL copybooks
- HBRHLQ.SHBRCOBS: COBOL sample code
- HBRHLQ.SHBREXEC: Installation specific exec
- HBRHLQ.SHBRINST: Install and configuration jobs
- HBRHLQ.SHBRJCL: Runtime sample jobs
- HBRHLQ.SHBRLOAD: Product modules
- HBRHLQ.SHBRPARM: Runtime configuration parameters
- HBRHLQ.SHBRPROC: Runtime JCL procedures
- HBRHLQ.SHBRWASC: Generated properties file for WebSphere Application Server configuration

2.6.2 Features

Depending on their role (architect, COBOL developer, QA tester, business user, policy manager, and so on), users are interested in various aspects of developing a business rule application and integrating it with the calling application on z/OS. Decision Server for z/OS offers these features:

- Business rule application development for z/OS

  To develop a business rule application for z/OS, you design business rules independently from the application logic. Using Rule Designer, you develop rule projects from which you extract a ruleset. Then, you create a contract between the application and the ruleset. An application can call the ruleset in a number of ways using various execution options.

- Validation of ruleset execution on z/OS

  With Rule Execution Server on WebSphere Application Server on z/OS, you can test the ruleset execution and simulate scenarios.

- Integrating ruleset execution into z/OS

  You have a number of options to execute a ruleset on z/OS. When you need a managed execution environment on z/OS, you must choose between WebSphere Application Server or zRule Execution Server for z/OS. In both cases, you can audit and monitor the

performance of the application using the administrative console. In a Java EE environment, you can also use Decision Warehouse.

Decision Server on z/OS offers multiple deployment options with and without a WebSphere Application Server base.

In the following sections, we focus on the business rule aspects of Decision Server.

2.6.3 Decision Server Rules

This section discusses Decision Server Rules.

Rule Designer

Rule Designer is used for the base rule authoring. This tooling is Eclipse-based and installed on a workstation. It cooperates with Decision Center. It is installed on a distributed operating system or on WebSphere Application Server installed on z/OS.

This split-platform configuration enables users to implement full business rule management system (BRMS) functionality with the ability to define a business object model from COBOL definitions, run and test rules on COBOL data, and call the rule engine from existing CICS and batch applications. Users, who want to eliminate the duplication of application functionality on z/OS and distributed applications, can consolidate applications and identify the rules that they can share between the two platforms.

The inclusion of COBOL management in Rule Designer enables users to author rules and develop object models that can be shared with COBOL and Java applications. Developers import a COBOL eXecutable Object Model from a copybook and create a Java eXecutable Object Model along with a marshaller project to provide the mapping between Java and COBOL.

You do not have to re-engineer or rewrite an entire COBOL application to start managing the business rules for your z/OS applications. You can base the scope of your rules on one, or a combination, of the following objects:

- A set of rules for a specific region, territory, or type of customer
- A process or subprocess within a z/OS application
- As a replacement for rules that might be hard-coded in your COBOL application

To write and review your rules, you create a rule project in the Rule perspective. And, you set up a rule authoring environment in Rule Designer.

This Rule Project can be synchronized with Decision Center and Rules Solution for Office to finally deploy executable rules with separate approaches.

Methods to invoke business rules from COBOL

When it comes to deployment, WebSphere Operational Decision Management for z/OS does not impose a rigid architectural or technical choice. Instead, it provides a set of options of which you can take advantage, depending on your strategy and architectural preferences.

This solution provides three options on WebSphere Decision Server for z/OS for deploying business rules on System z, all of them providing the following values:

- Reduced risk, disruption, cost, and time to implement change
- Better visibility and maintainability of decision logic
- Improved decision logic reuse across applications

Rule Execution Server on WebSphere Application Server for z/OS

This option brings the full power of WebSphere Application Server for z/OS to the rule execution on z/OS. Full high availability and scalability are provided by the underlying application server, and the full suite of decision management services is available. Because of the new ability to consume COBOL data structures directly, the Rule Execution Server environment can be easily integrated with existing COBOL applications. This integration occurs through the use of technologies, such as the WebSphere Optimized Local Adapter or WebSphere MQ.

zRule Execution Server for z/OS

This option provides more local integration with existing COBOL applications. A supplied COBOL stub program provides an interface that can call COBOL directly into the rule execution. This rule execution environment provides COBOL applications with access to the full rule-authoring constructs through three available runtime options:

- Stand-alone mode provides a rule execution address space that can be invoked from existing COBOL applications by way of the supplied callable stub.
- In IBM CICS Transaction Server for z/OS V4.1 and V4.2, an optimized local execution option uses the new Java Virtual Machine (JVM) server environment to allow execution within the CICS region.
- An option of a COBOL application with the HBR API and zRule Execution Server Runtime (RT) and the zRule Execution Server Console.

COBOL code generation

This option is for clients who want to retain their existing application architecture and manage their business decisions within COBOL application code. However, they want a cleaner and more manageable form than through standard application development methods. The rules in a COBOL application can be incrementally migrated to a central business rule repository for external management, directly by business users. Then, the rules can generated back into COBOL code to be inserted into and called directly from the application. This option can also be a first step toward the incremental modernization of applications.

Decision Service is deployed as COBOL source code. It offers a pure COBOL approach but without giving full agility as the Rule Execution Server on WebSphere Application Server for z/OS or zRule Execution Server for z/OS options. The difference is the compiled and linked code. Additionally, this source generation option does not offer support for Decision Validation Service (DVS) and is limited to the Sequential algorithm. See Figure 2-5 on page 22.

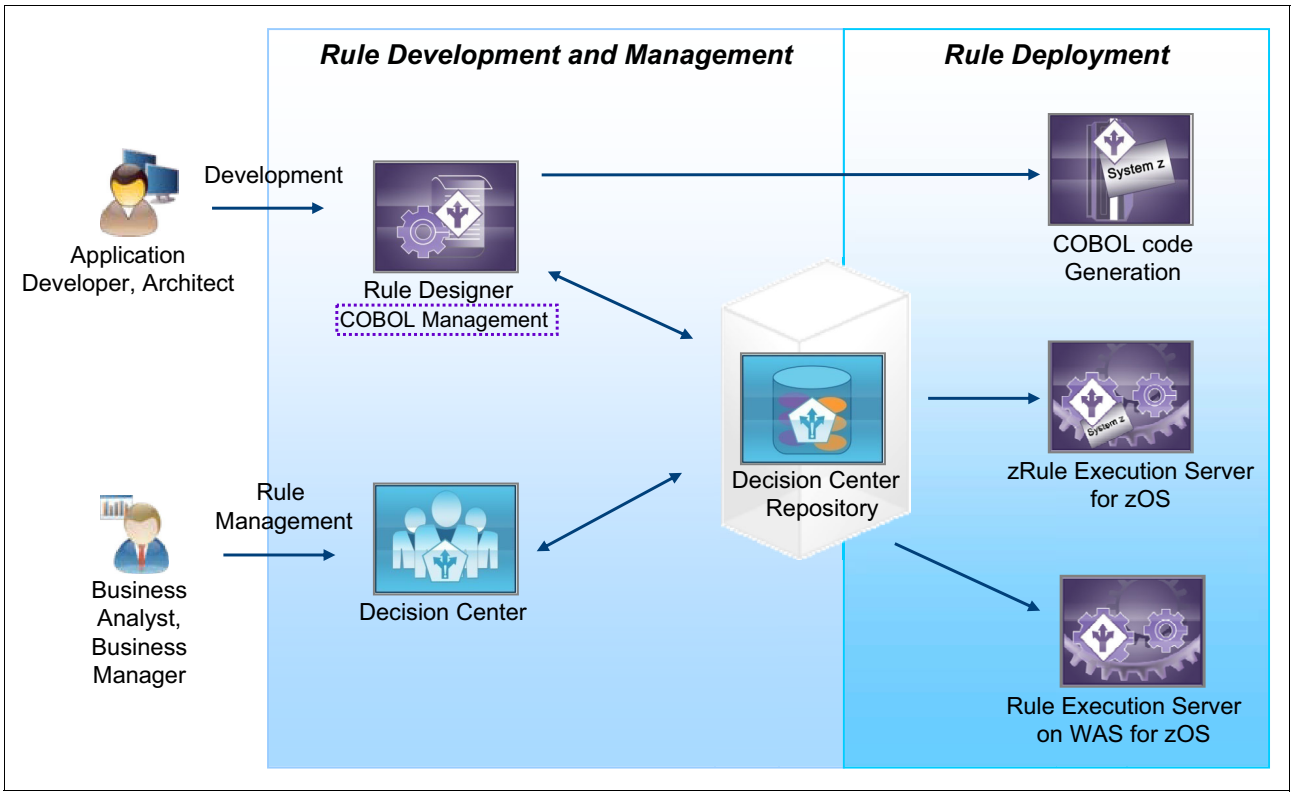


Figure 2-5 WebSphere Operational Decision Management for z/OS: Business rules options

A feature comparison between zRule Execution Server for z/OS and WebSphere Application Server is shown in Table 2-2.

Table 2-2 Feature comparison between zRule Execution Server for z/OS and WebSphere Application Server

Feature	zRule Execution Server for z/OS	WebSphere Application Server for zRule Execution Server for z/OS	COBOL source generation
Execution from Java	No	Yes	No
Execution from COBOL	Yes	Yes through WebSphere Optimized Local Adapter	Yes
OOTB COBOL marshalling	Yes	No	N/A
Testing	No	Yes	No
Simulation	No	Yes	No
Hosted transparent decision services	No	Yes	No

Deployment of runtime options across environments

Figure 2-6 on page 23 shows the run times to invoke business rules from a COBOL application.

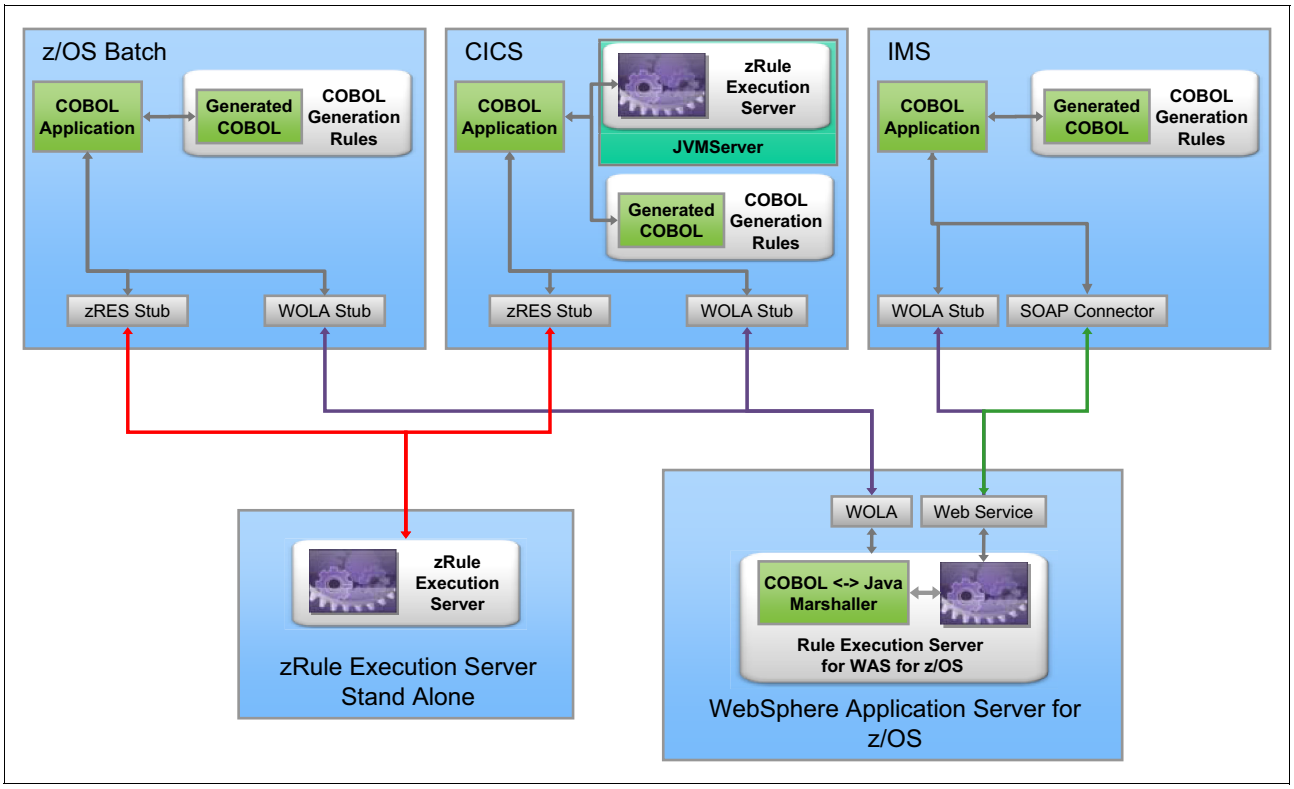


Figure 2-6 Runtime options to invoke business rules from a COBOL application

2.6.4 Decision Server Events

This section discusses Decision Server Events.

Event Designer

Event Designer is a Decision Server Events component that supports the definition of the metadata layer that is required for business event processing (BEP). You can use Event Designer to create all the building blocks for your application, including events, business objects, actions, and event rules. Event Designer is based on Eclipse.

Event Designer brings an Eclipse event perspective, in which you manage event projects.

Event run time for WebSphere Application Server on z/OS

The event run time consists of a number of components that manage the real-time business event coordination that was defined during application development. The event run time consists of the following components (Figure 2-7 on page 24):

- The Decision Server Events application for BEP, which is called wberuntimeear
- A Java Message Service (JMS) message queue that is managed by a message queue server, such as the WebSphere Application Server Network Deployment service integration bus or WebSphere MQ
- A database manager to serve as a message store for persistent messages
- WebSphere Application Server Network Deployment
- A relational database to contain the event run time
- Event connectors and action connectors for touchpoint systems

- Appropriate Java Database Connectivity (JDBC) drivers for the event runtime database manager and any databases accessed in a Decision Server Events application

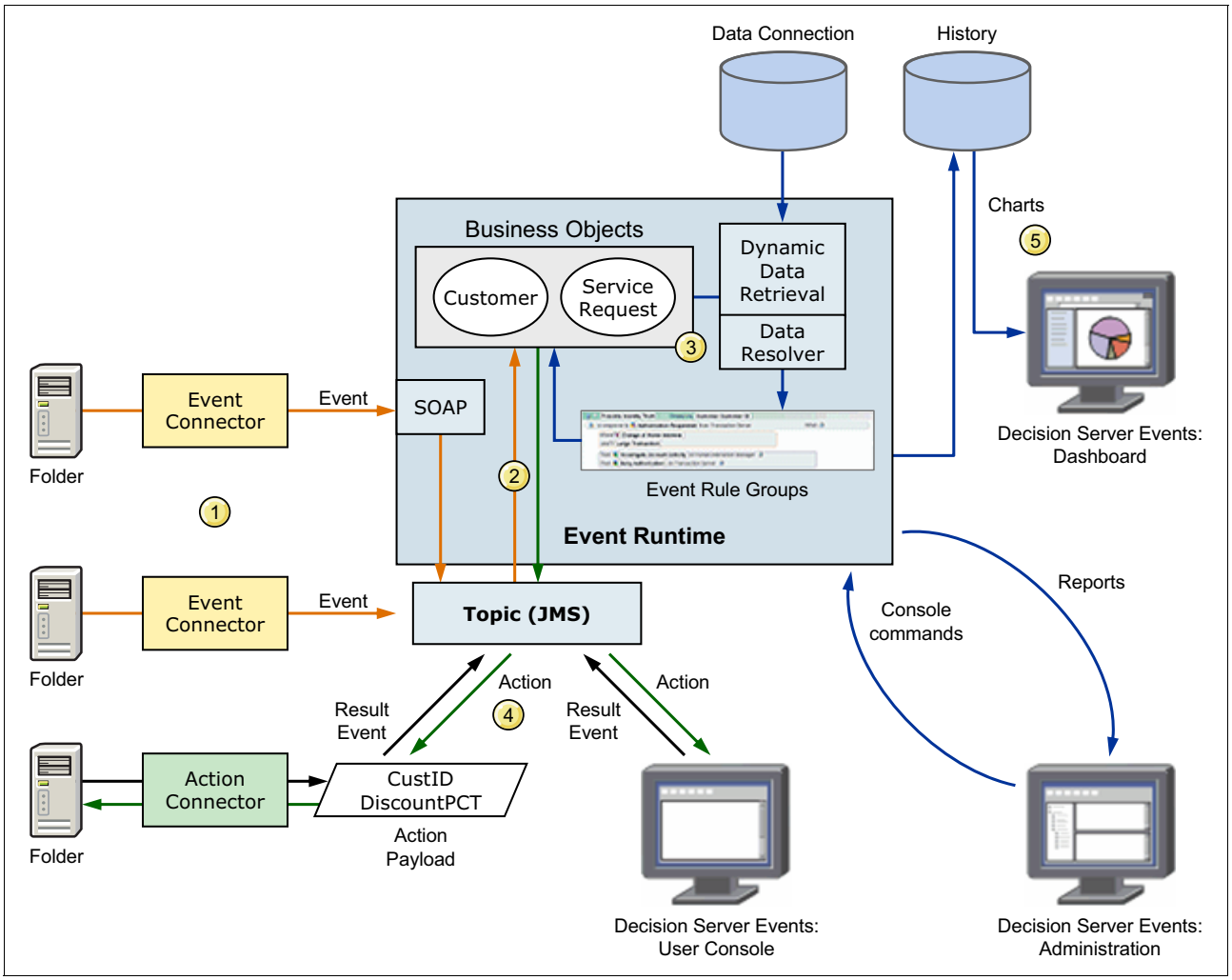


Figure 2-7 WebSphere Operational Decision Management Decision Server Event

The event run time manages the real-time business event coordination that was defined during application development (the following numbers correspond to the numbers that are shown in Figure 2-7):

1. When an event occurs in a touchpoint system that potentially requires one or more actions in other touchpoint systems, the relevant data (field name, field type, and value), which is called an *event payload*, is passed to the JMS topic by using the connector of the touchpoint system. Web services environments that employ SOAP to package messages and other protocols, such as HTTP, can direct those messages through the JMS queue.
2. The event run time retrieves the message from the JMS queue and populates the appropriate business objects with the values that are contained in the event payload.
3. The event run time identifies the event rules that reference the event, and determines whether any filters exist that require further evaluation.

   If a rule includes a condition, the event run time evaluates that condition to determine whether the conditions for an action are met. For complex event processing, this evaluation includes determining whether referenced events or actions occurred as described in the conditions. If any values are missing, the event run time attempts to

retrieve the missing information from an external data connection. The action is triggered only if the condition is true.

If the rule does not include any conditions, the specified action is triggered.

4. The event run time passes the relevant data (field name, field type, and value) associated with the action from business objects as an *action payload* to the outbound message queue, where it is picked up by the appropriate connector. The connector pushes the data to the appropriate touchpoint systems, initiating the appropriate activity. The connector might return a result back to the JMS queue, where it is retrieved by the event run time as a new event and processed appropriately.

   If the action requires human intervention, it is directed to the user console, where you can access it. When you respond to the information that is displayed, the response is sent back to the JMS queue as a result event, where it is retrieved by the event run time as a new event and processed appropriately.

5. The history for events, actions, and filters used in event rule group evaluation is stored in a History database manager. This information is retrieved, formatted, and displayed as charts in real time in Business Space by the Event Chart widget.

3

Getting started with business rules

This chapter discusses the use of business rules on System z and includes the following topics:

- Overview
- Getting started from a COBOL copybook
- Getting started from an existing rule project

© Copyright IBM Corp. 2012. All rights reserved.

3.1 Overview

This section provides an overview of the business scenario and the related models that are used in this chapter.

3.1.1 Business scenario

The business scenario that is used in this book describes a fictitious auto insurance company that has a solution in place for validating an insurance application for its customers. The company wants to manage and share this validation logic with other business applications on System z.

The status of the in-place application and the way in which a company wants the business rules to be executed on System z determine the approach that is chosen. This chapter uses the following typical approaches:

- Getting started from a COBOL copybook
- Getting started from an existing Java-based rules project

3.1.2 Business model

This section describes a business model that represents an insurance quote application. This model is simplified as one insurance quote request and one insurance quote response.

Insurance quote request

The insurance quote request includes the following information:

- Driver

  This information includes personal information about the driver to assess the risk, such as age, address, and license status.

- Vehicle

  This information includes the make, model, year, and vehicle identification number, together with a categorization of the vehicle type and a vehicle value. The insurance discount policies differ, based on the vehicle type.

Insurance quote response

The response to an insurance request includes the following information:

- The validation status of this insurance quote
- The validation message of this insurance quote
- The pricing and discount information of this insurance quote

3.1.3 Scenario rule model

The rules are designed to validate a customer's eligibility for the quote application. These sets of rules validate the customer's age or accident history and provide the validation result and possible reasons.

Both scenarios use the following rules:

- A maximum or minimum age rule, called *MaxiMinimumAge,* validates that the age of customer is between lower and upper age limits.
- A number of accidents rule, called *NumberOfAccidents*, validates that the number of accidents of the customer is below the set upper limits.

3.1.4 Project structure of a business rule on z/OS

This section illustrates a common project structure for business rule execution on zRule Execution Server for z/OS. The following typical artifacts are used in zRule Execution Server for z/OS projects:

- The *rule project*, which is used to design, debug, and manage your business rule
- The *Java Execution Module (XOM)*, which is used to create a rule project and which is deployed as a Java archive (JAR) resource for rule execution at run time
- The *marshaller*, which handles the conversion between COBOL data items and Java data items at run time

> **Important:** The marshaller is a generated artifact. Never modify it yourself.

- The *RuleApp project*, which is used to deploy business rules to the runtime configuration of the zRule Execution Server

3.2 Getting started from a COBOL copybook

This section includes detailed instructions about creating a rule application on z/OS that is started from a COBOL copybook.

3.2.1 Scenario overview

In this scenario, an insurance company has a large COBOL application that runs on z/OS. This COBOL application validates insurance applications. The company wants to manage the logic codes that are scattered throughout the COBOL application and share them with other business applications on System z. The company decides to migrate and manage the business logic as a business rule application on System z.

In an actual scenario, you identify or create a COBOL copybook that contains the COBOL data items that are required for the business rules. We show a sample COBOL copybook (INSDEMO) in Example 3-1, which is designed for the first rule application on System z in this scenario.

Example 3-1 Sample COBOL copybook [INSDEMO.cpy]

```
01  REQUEST.
      05  DRIVER.
    10  FIRST-NAME              PIC  X(20).
    10  LAST-NAME               PIC  X(20).
        10  ZIPCODE                PIC  X(8).
        10  HOUSE-NUM               PIC  9(8).
        10  AGE                     PIC  9(2) USAGE COMP-3.
        10  LIC-DATE                PIC  X(8).
```

```
      10  LIC-STATUS              PIC  X.
      10  NUMBER-ACCIDENTS        PIC  99.
    05  VEHICLE.
      10  VEC-ID                  PIC  X(15).
      10  MAKE                    PIC  X(20).
      10  MODEL                   PIC  X(20).
      10  VEC-VALUE               USAGE COMP-1.
      10  VEC-TYPE                PIC  X(2).
          88 SUV     VALUE 'SU'.
          88 SEDAN   VALUE 'SD'.
          88 PICKUP  VALUE 'PU'.
01  RESPONSE.
    05  APPROVED                  PIC  X.
    05  BASE-PRICE                USAGE COMP-2.
    05  DIS-PRICE                 USAGE COMP-2.
    05  MSG-COUNT                 PIC 9(5)  VALUE 0.
    05  MESSAGES                  PIC  X(100)
                                   OCCURS 0 TO 100 TIMES
                                   DEPENDING ON MSG-COUNT.
```

Additional resources: You can find the `INSDEMO.cpy` copybook file in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. Refer to Appendix A, “Additional material” on page 259.

3.2.2 Creating a rule project

You can create a rule project in Rule Designer. A *rule project* enables you to manage, build, and debug the items that make up the business logic of your application.

Follow these steps to create the rule project in Rule Designer:

1. Click **File** → **New** → **Rule Project**. Then, select **Standard Rule Project** and click **Next**.
2. In the Project name field, type `insurance-rules`, as shown in Figure 3-1. Then, click **Finish**.

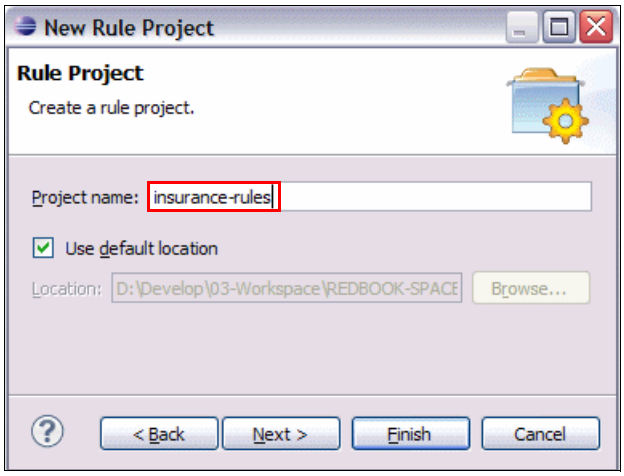


Figure 3-1 Type rule project name

The new rule project is created in Rule Designer, as shown in Figure 3-2. For now, the rule project contains only empty folders.

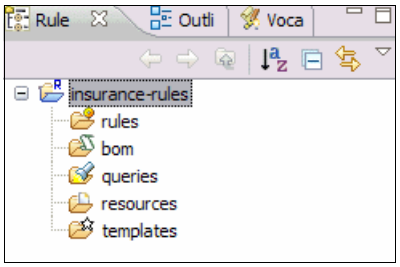


Figure 3-2 New rule project in the Rule Explorer view

3.2.3 Creating COBOL XOM from a COBOL copybook

To execute rules in a COBOL application, you generate the COBOL XOM from a COBOL copybook. A COBOL XOM provides the necessary COBOL-to-Java mapping so that you can create and execute your rules from a COBOL application.

To use the Rule Project Map to guide you through the COBOL XOM generation, follow these steps:

1. In the Design part of the Rule Project Map tab, click **Import XOM**, as shown in Figure 3-3.

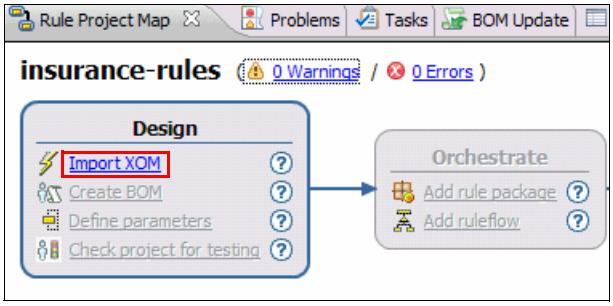


Figure 3-3 Import XOM in Rule Project Map

2. In the Import XOM page, choose **COBOL Execution Object Model**, as shown in Figure 3-4, and click **OK**.

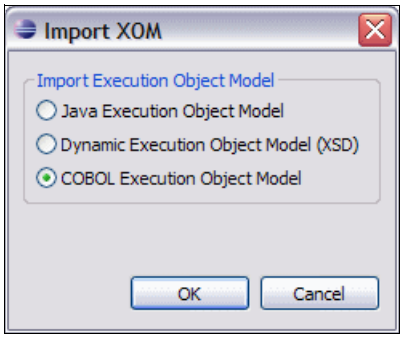


Figure 3-4 Choose COBOL Execution Object Model

3. On the Import COBOL XOM page, in the COBOL XOM name field, enter `insurance`, as shown in Figure 3-5. Then, click **File System**, and select the **INSDEMO.cpy** copybook.

> **Additional resources:** You can find the `INSDEMO.cpy` copybook file in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. Refer to Appendix A, “Additional material” on page 259.

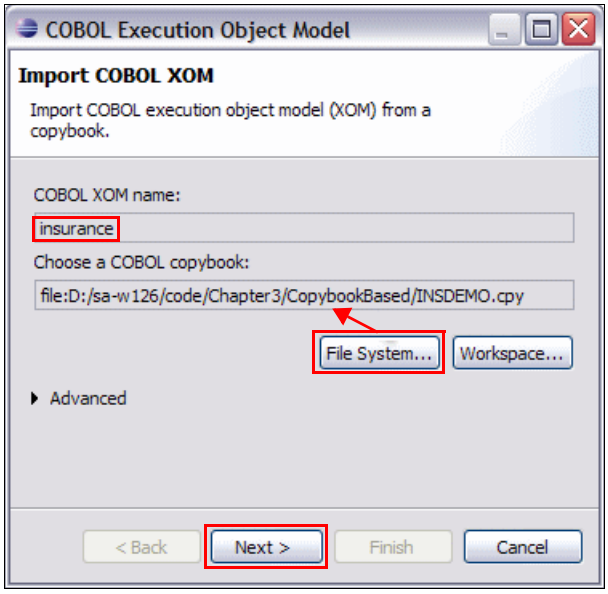


Figure 3-5 Select COBOL copybook

4. Click **Next** to display a summary of the default Java types and business object model (BOM) attributes that are derived from each COBOL item in the copybook.
5. Now, use the type converter to map two COBOL string items (PIC X) to Java Date or Boolean type. Change LIC-DATE from type String to Date, using the type converter, in the following steps:
   a. Expand the **REQUEST** item. Then, right-click the row that contains the **LIC-DATE** data item, and click **Add Converter**, as shown in Figure 3-6 on page 33.

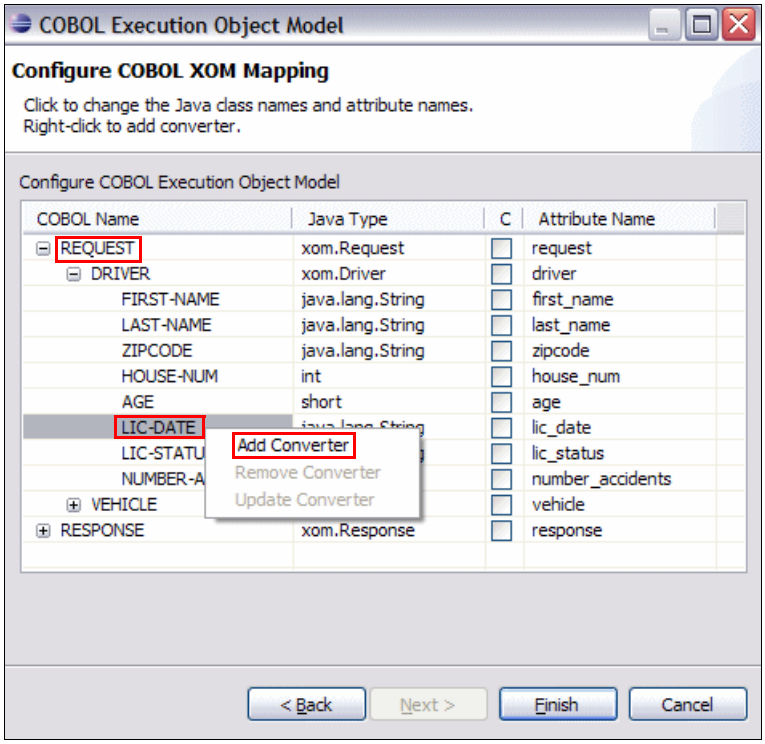


Figure 3-6 Add converter for LIC-DATE item

b. On the Configure Converter Settings page, select **Built-in String to Date Converter**. Then, for the Date format field, enter `yyyyMMdd`, as shown in Figure 3-7. Click **OK**.

LIC-DATE: Use *yyyyMMdd* here to parse the LIC-DATE item value, such as `20110908`.

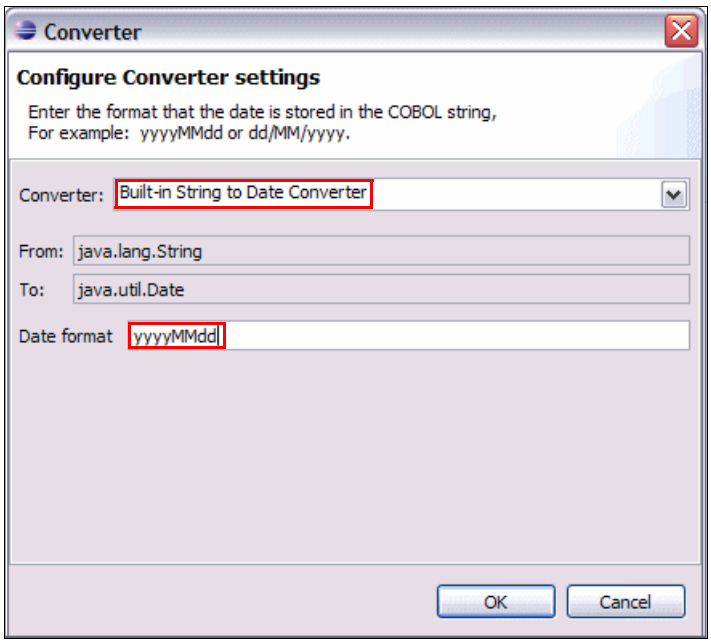


Figure 3-7 Configures the date converter

6. Next, change APPROVED from type String to boolean, using the type converter:
   a. Expand the **RESPONSE** item. Then, right-click the row that contains the **APPROVED** data item and click **Add Converter**.
   b. On the Configure Converter Settings page, select **Built-in String to boolean Converter**.
   c. For the True value field, type `T` and for the False value field, type `F`, as shown in Figure 3-8. Then, click **OK**.

> **Values:** The *T* and *F* values are COBOL values that represent True and False. You can also customize these value as Y/N, YES/NO, and so on.

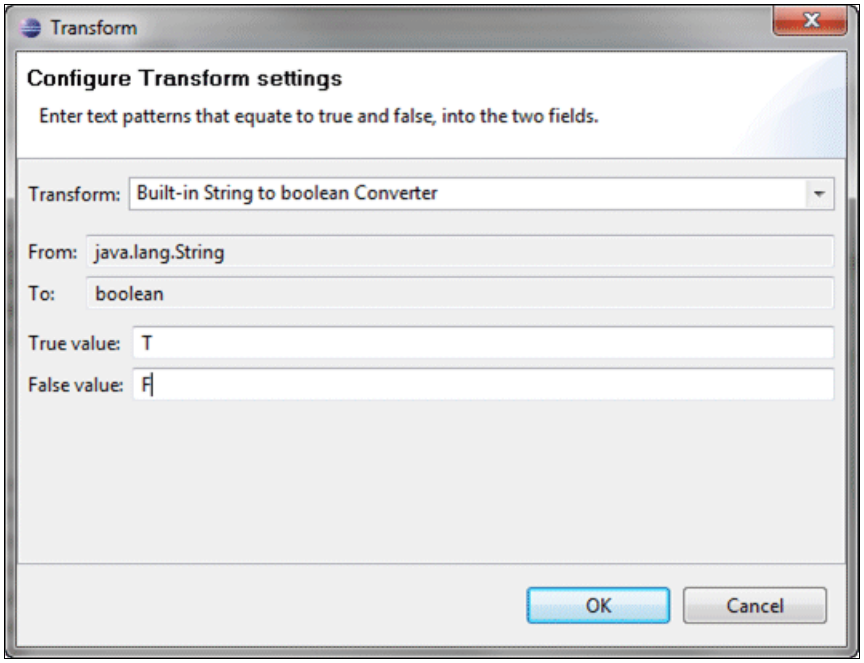


Figure 3-8 Configures boolean converter

7. Click **Finish** to create the COBOL XOM, as shown in Figure 3-9.

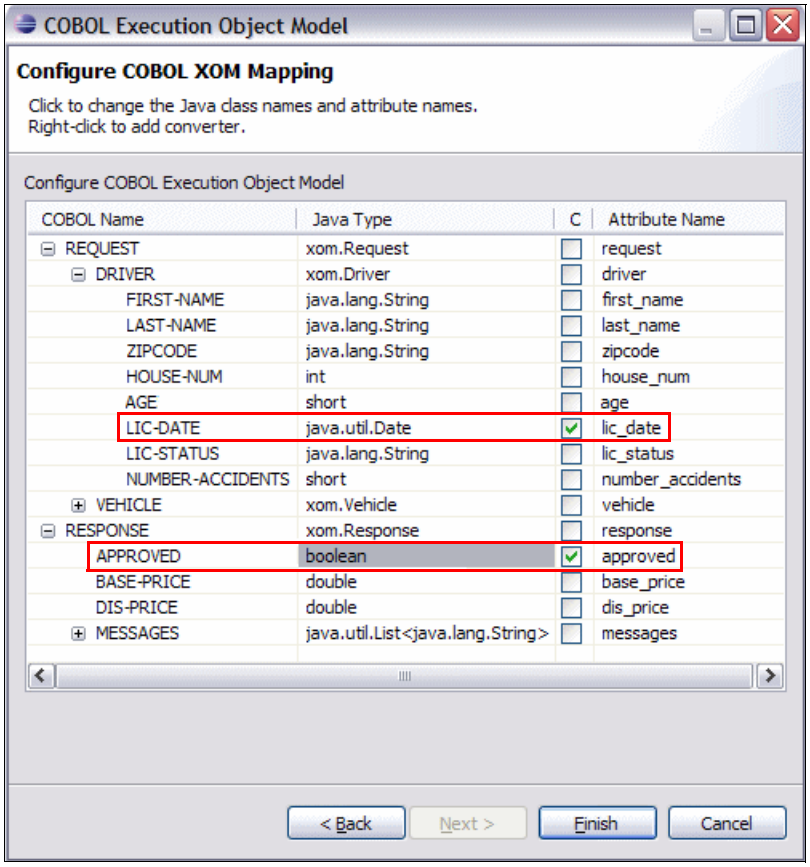


Figure 3-9 Finish data type configuration

8. Click **OK** to close the Properties window, as shown in Figure 3-10.

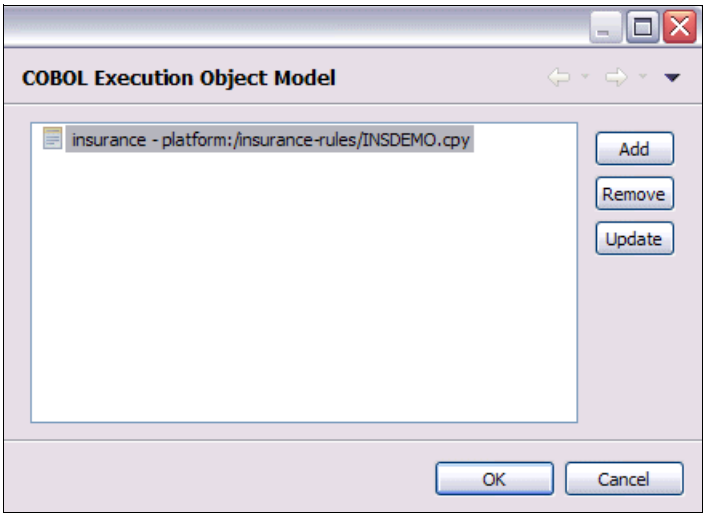


Figure 3-10 Generate COBOL XOM

The following artifacts are created, as indicated in Figure 3-11:

- Java XOM project: `insurance-xom`
- Mashaller project: `insurance-marshaller`
- COBOL XOM configuration file: `CobolXomConfig.xml`

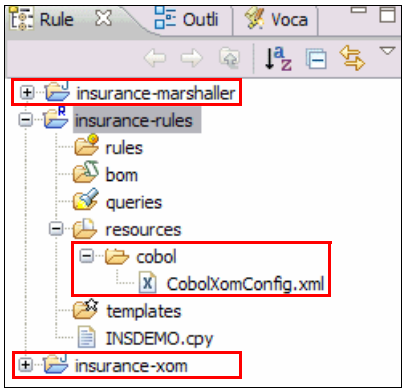


Figure 3-11 View COBOL XOM artifacts

Important: The Java XOM and marshaller projects are generated artifacts. Do *not* change these projects manually.

3.2.4 Creating a business object model from the Java XOM

The *business object model (BOM)* is a business layer that is used to author business rules. This section describes how to create a BOM in Rule Designer that is based on the Java XOM that you created in the previous section.

Follow these steps to create a BOM from the COBOL XOM:

1. In the Design part of the Rule Project Map tab, click **Create BOM**, as shown in Figure 3-12.

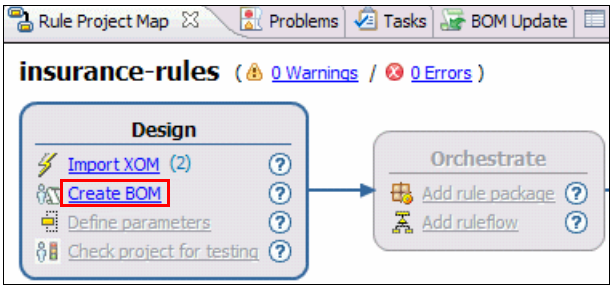


Figure 3-12 Select Create BOM from the Rule Project Map

2. In the New BOM Entry wizard, in the Name field, accept the default name for the BOM entry. In this scenario, the default name is `model`. Ensure that the “Create a BOM entry from a XOM” option is selected, as indicated in Figure 3-13, and then click **Next**.

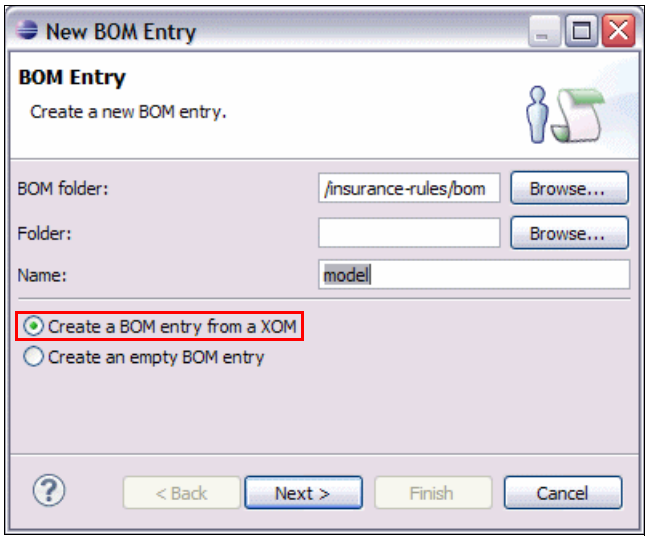


Figure 3-13 Create a BOM entry from a Java XOM

3. On the Browse XOM page, in the “Choose a XOM entry” field, click **Browse XOM**, select **insurance-xom**, as indicated in Figure 3-14, and then click **OK**.

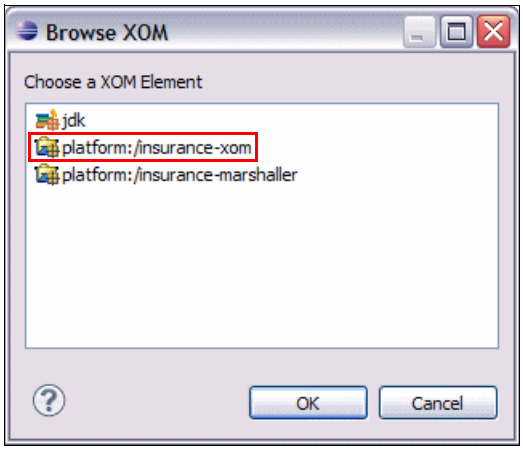


Figure 3-14 Select generated Java XOM

Important: Make sure that you select the XOM project, *not* the marshaller project.

4. In the “Select classes” field of the New BOM Entry window, select the **XOM** package. When you select the package, you automatically select all the classes that it contains, as indicated in Figure 3-15.

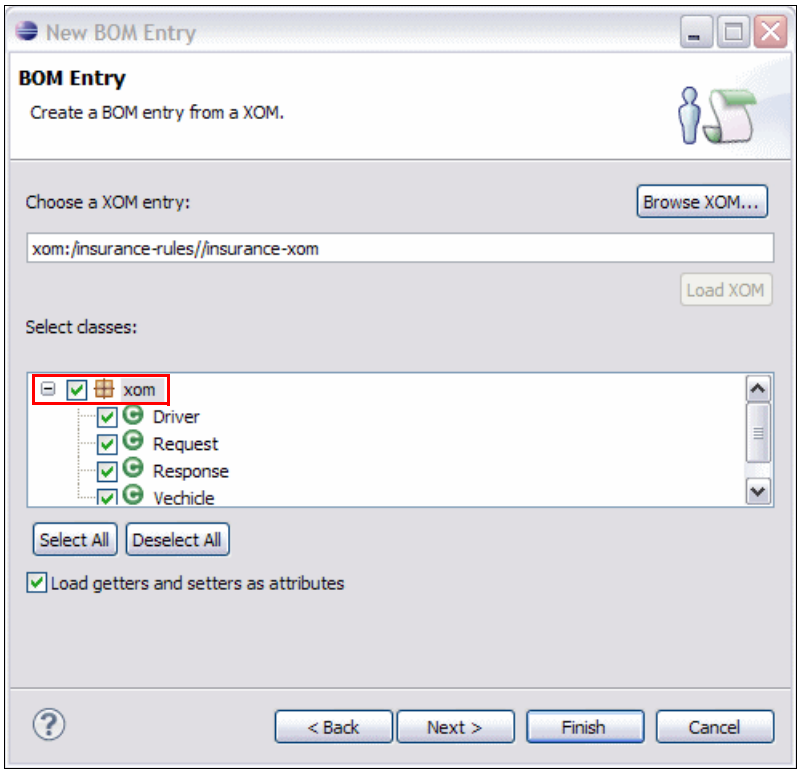


Figure 3-15 Select package to import all classes

5. Click **Finish**. In the Rule Explorer view, the `bom` folder contains a new BOM entry model, as shown in Figure 3-16.

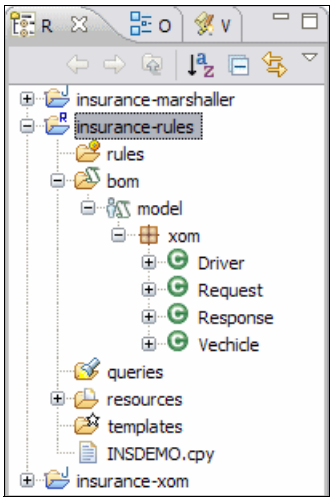


Figure 3-16 View generated BOM

6. View the generated BOM and its verbalization:
   a. In the Rule Explorer view, double-click **bom** → **model** to open the BOM Editor.

b. Then, in the BOM Editor, expand the **xom** package to view the generated BOM, as shown in Figure 3-17.

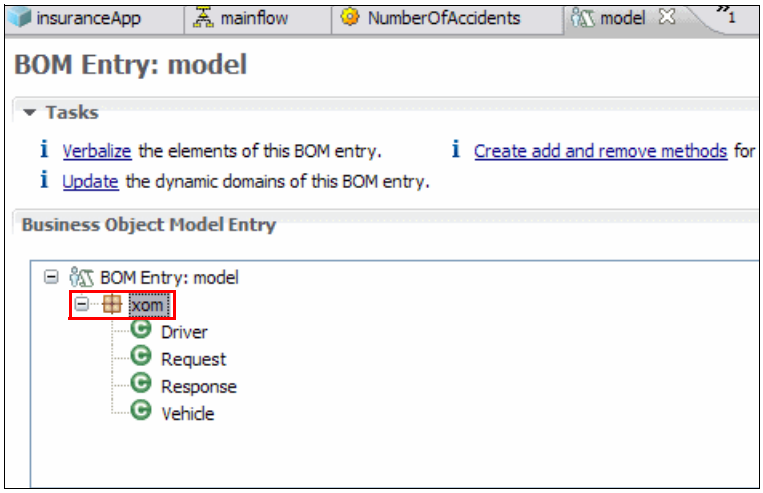


Figure 3-17 View the generated BOM

c. In Figure 3-17, double-click the **Driver** class to view the default class verbalization. The resulting Class Verbalization window is shown in Figure 3-18.

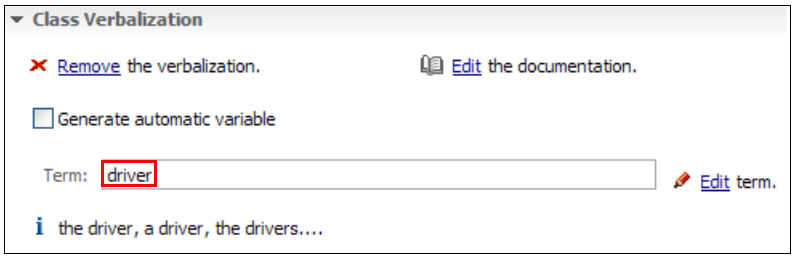


Figure 3-18 View default verbalization

Language: The default verbalization is in English. If you are working in a localized version of Rule Designer, you can verbalize the BOM classes in the language of your locale.

3.2.5 Declaring ruleset parameters

Ruleset parameters provide the means to exchange data between a COBOL application and the rule application. You define ruleset parameters by name, type, and direction.

In this sample, you decide on the status of an insurance request and response, so that you create ruleset parameters for the Request and Response classes. You use the IN direction for the request parameter. The value of the request parameter is provided as input from the COBOL client application on execution. The direction for the response parameter must be IN_OUT. The value of the request parameter is set by the IN value passed by the client and then updated by the engine on the way OUT. The updated value is returned to the client.

Important: You *cannot* use the OUT parameter direction with zRule Execution Server for z/OS, because COBOL programs do not support memory allocation dynamically.

Follow these steps to declare ruleset parameters:

1. In the Design part of the Rule Project Map tab, click **Define parameters**, as shown in Figure 3-19.

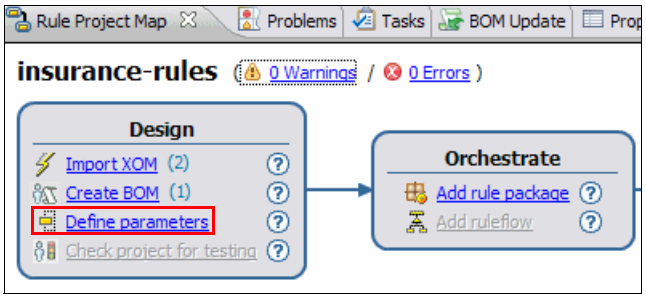


Figure 3-19 Select Define parameters

2. On the Ruleset Parameters page, select **Enable type check for COBOL XOM**.
3. To define a request parameter, click **Add**. Then, change the following default values, as shown in Figure 3-20:
   - In the Name column, type `request`.
   - In the Type column, click the ellipsis (**...**) button on the right of the cell, and choose **Request**. The `xom.Request` entry is entered in the cell automatically.
   - In the Direction column, chose the **IN** direction.
   - In the Verbalization column, type `the insurance request`.

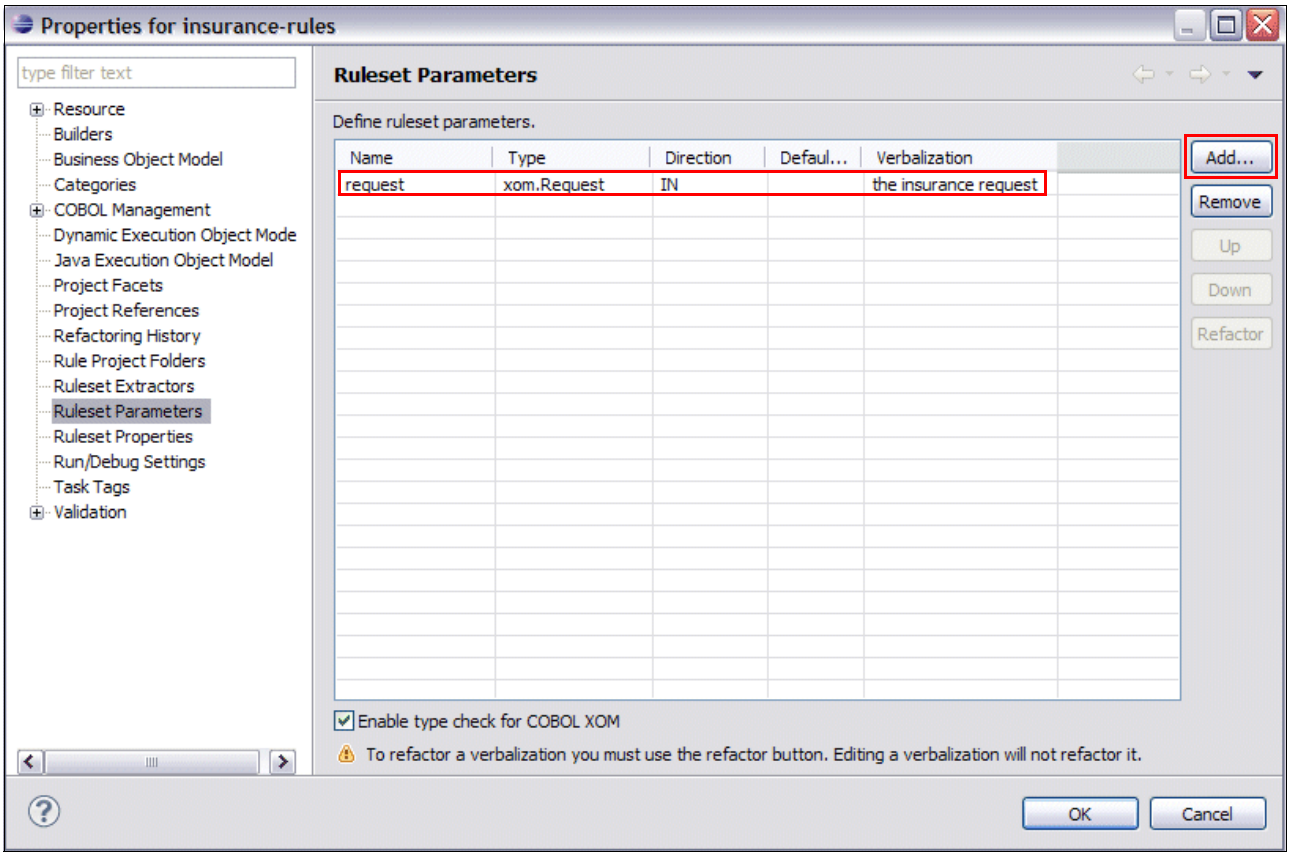


Figure 3-20 Add the ruleset parameter for request

4. To define the response parameter, click **Add**. Then change the following default values, as shown in Figure 3-21:
   - In the Name column, type `response`.
   - In the Type column, choose **Response**. The `xom.Response` entry is added to the cell automatically.
   - In the Direction column, chose the **IN_OUT** direction.
   - In the Verbalization column, type `the insurance response`.

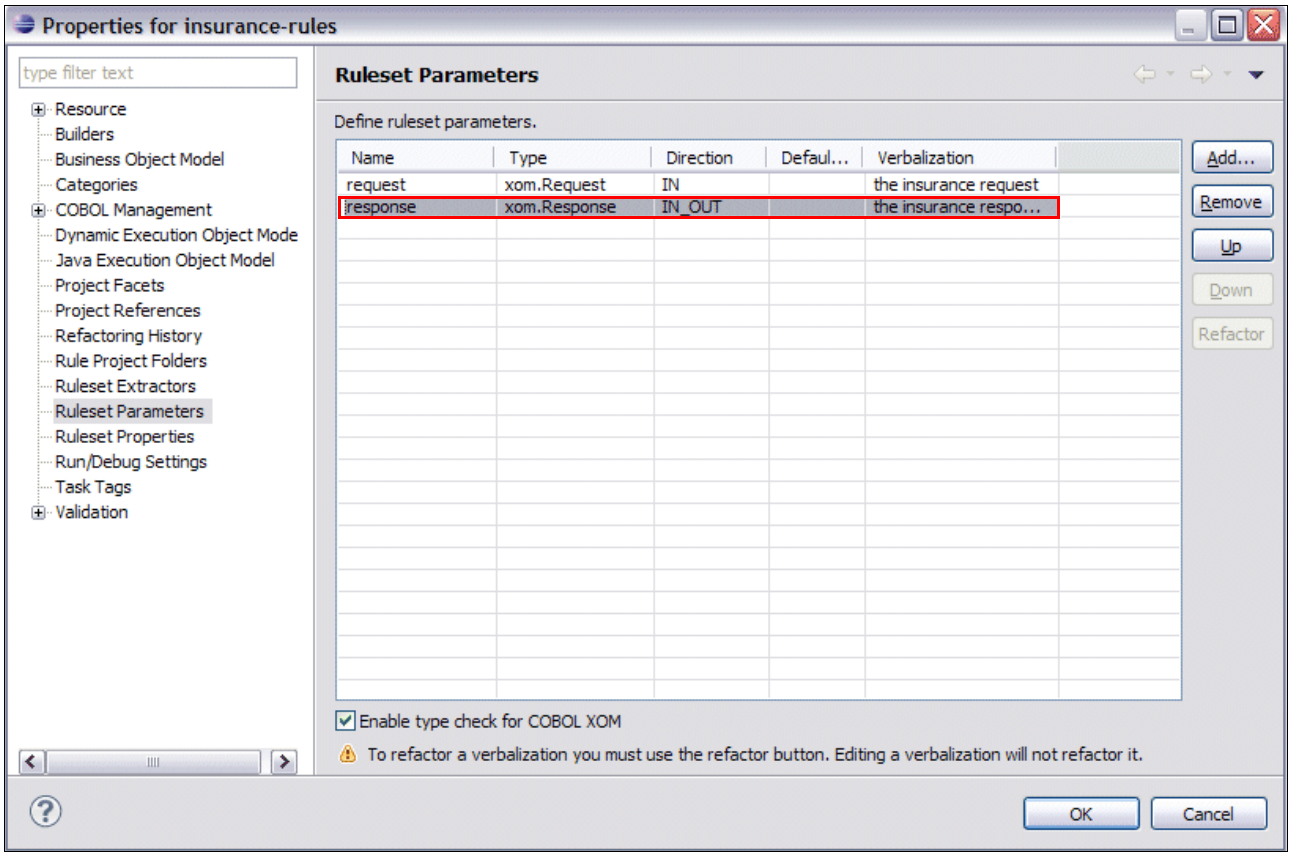


Figure 3-21 Add ruleset parameter for response

5. Click **OK** to close the Properties for insurance-rules dialog box.

3.2.6 Adding BOM methods and mapping them to the XOM

You use methods to specify conditions and actions in your rules. You create methods in Rule Designer. When you add methods to the BOM, you use BOM to XOM mapping in the BOM Editor to implement the method.

> **Important:** You *cannot* map the BOM method to a Java XOM method, because you must not change the XOM.

This section describes how to add the following BOM methods:

- addMessage: Defines what is needed to pass information from the rules
- reject: Identifies whether the insurance request was rejected

Adding the addMessage method

To add the addMessage method, follow these steps:

1. In the Outline view, expand the **model** package, and then double-click the **Response** class, as shown in Figure 3-22.

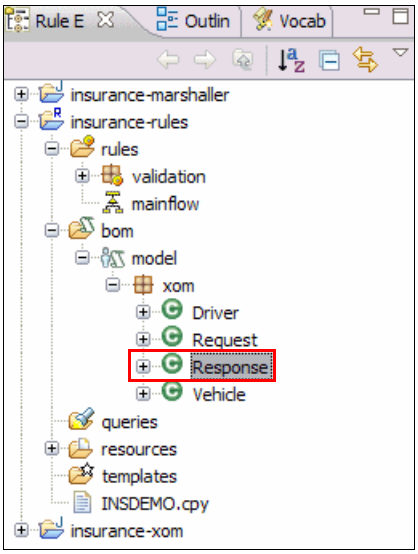


Figure 3-22 Selecting the Response class

2. On the Class Response page of the BOM editor, to the right of the Members section, click **New**, as shown in Figure 3-23.

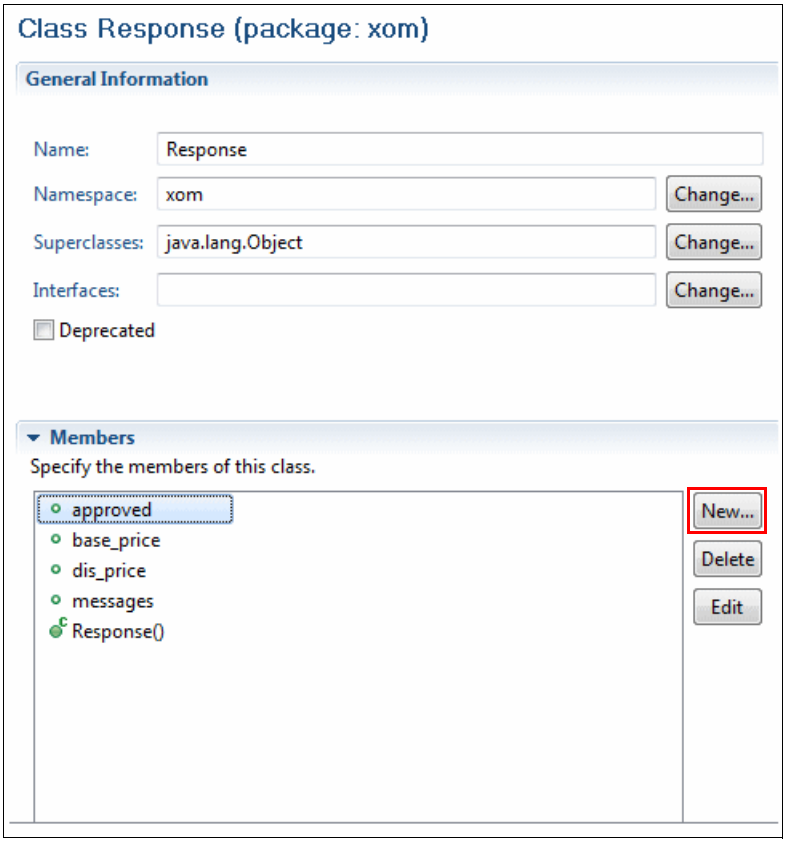


Figure 3-23 New member

3. In the New Member window, enter this information:
   - For the Type, click **Method**.
   - For the Name, type `addMessage`.
   - For the Type, enter `void`.
   - Click **Add**, as shown in Figure 3-24 on page 44.

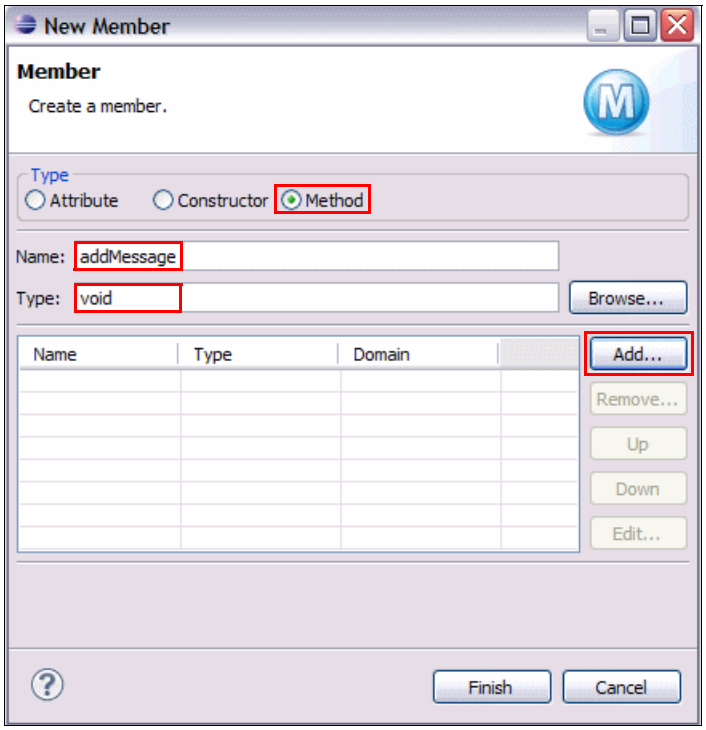


Figure 3-24 Create new method for addMessage

4. In the Method Argument window, enter this information:
   - For the Name, type `msg`.
   - For the Type, enter `java.lang.String`.
   - Click **OK**, as shown in Figure 3-25. Then, click **Finish**.

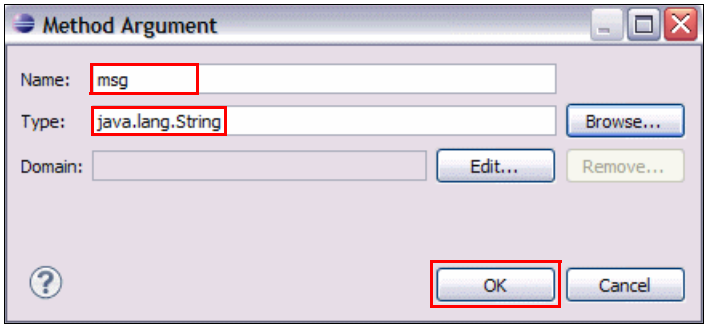


Figure 3-25 Add the method argument

5. On the Class page of the BOM editor, the Members list now includes the addMessage(String) method, as shown in Figure 3-26.

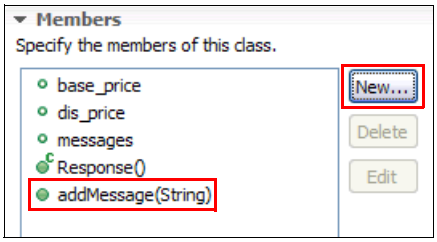


Figure 3-26 List addMessage method

6. Double-click the **addMessage** method. In the Member Verbalization section of the BOM editor, click **Create** to view the default verbalization, as shown in Figure 3-27.

Figure 3-27 Create verbalization

7. The default verbalization of the addMessage class now displays. Keep the default verbalization of “add {0} to the messages of {this}”, as shown in Figure 3-28.

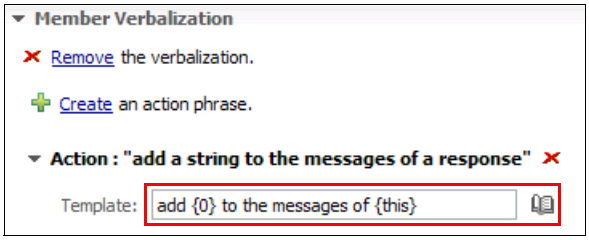


Figure 3-28 Keep the default verbalization

8. Scroll down to the BOM to XOM Mapping section of the BOM editor, and then expand **BOM to XOM Mapping** to activate the BOM to XOM Mapping editor, as indicated in Figure 3-29.

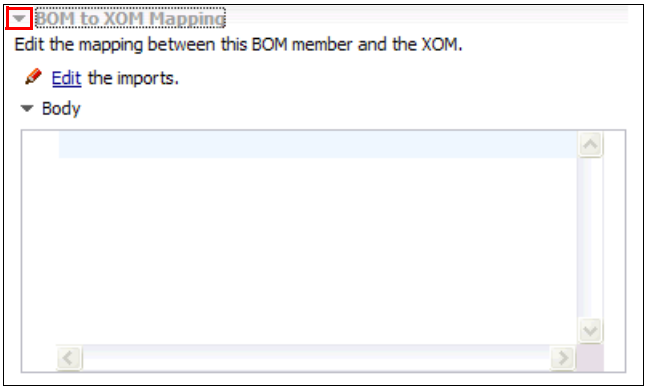


Figure 3-29 Activate BOM to XOM Mapping editor

9. Type the following Java code, as shown in Figure 3-30:

```
this.messages.add(msg) ;
```

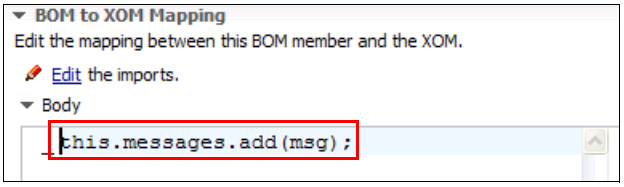


Figure 3-30 Add method implementation

10. Save your work.

Adding the reject method

To add the reject method, follow these steps:

1. Double-click the **Response** class in the Rule Explorer. Then, under Members, click **New**, as indicated in Figure 3-31.

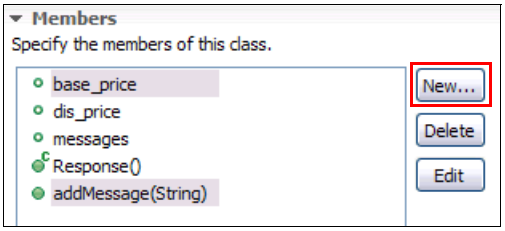


Figure 3-31 New reject method

2. In the New Member window, as shown in Figure 3-32 on page 47, enter this information:
   - For the Type, click **Method**.
   - For the Name, type `reject`.
   - For the Type, type `void`.
   - Click **Finish**.

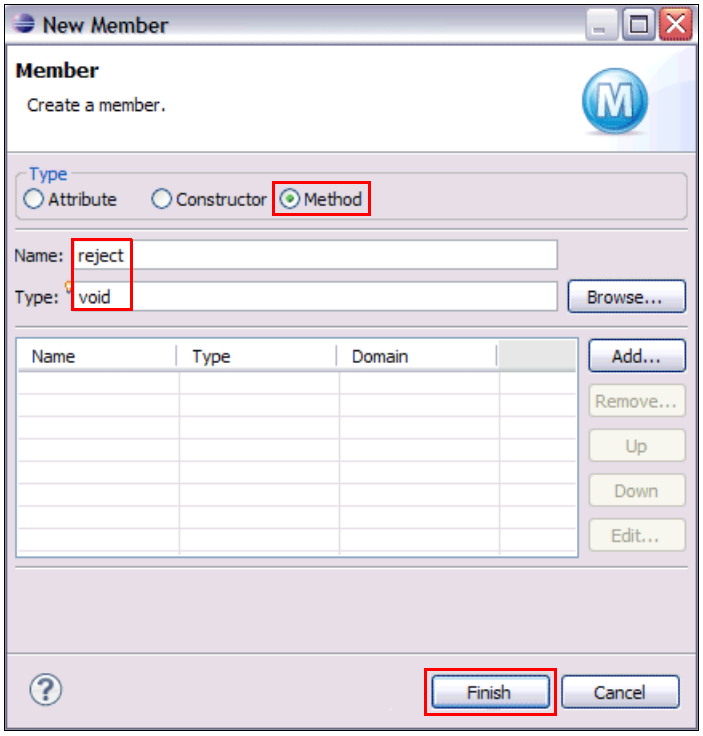


Figure 3-32 Define the method argument

3. Create the default verbalization for the reject method. Click **Create**, and accept the default verbalization of “reject {this}”, as shown in Figure 3-33.

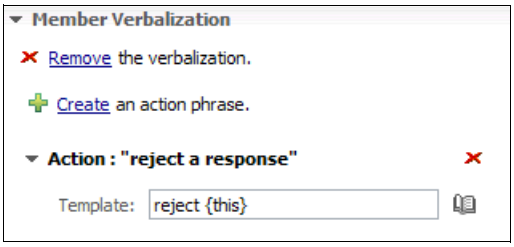


Figure 3-33 Define verbalization for the reject method

4. Define the BOM to XOM Mapping, as shown in Figure 3-34:
   - Click **Edit**.
   - Type `this.approved = false;` and save your work.

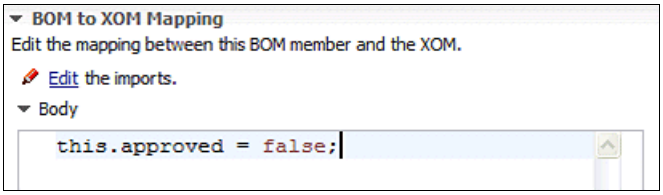


Figure 3-34 Implement reject method

If you look in the Rule Explorer, these members are now present in their classes, as shown in Figure 3-35.

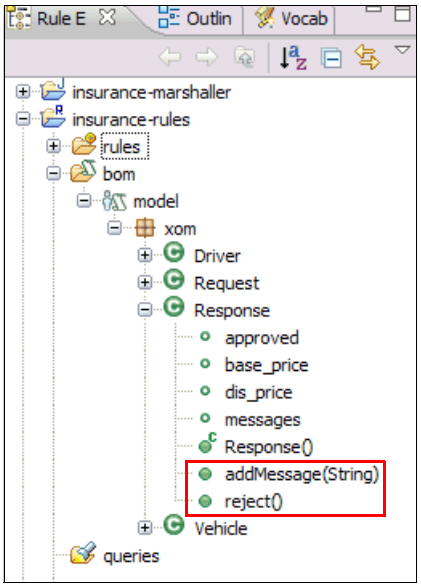


Figure 3-35 View new BOM method

3.2.7 Creating the ruleflow

Before writing the rules, you orchestrate how the rules execute. You control the order in which rules are executed by using *ruleflows*. When defining the flow of execution, you organize rules into packages that contain related rules. This section explains how to create a package that relates to the validation rules.

Follow these steps to create a ruleflow:

1. In Rule Designer, in the Orchestrate part of the Rule Project Map, click **Add rule package**, as indicated in Figure 3-36.

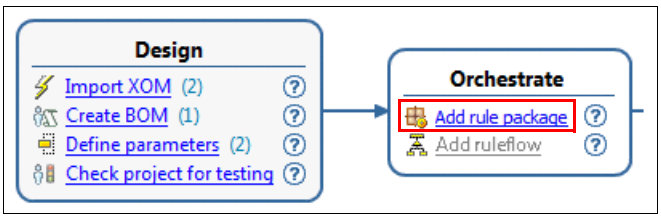


Figure 3-36 Add new rule package

2. In the New Rule Package wizard, in the Package field, type `validation`, and then click **Finish**, as shown in Figure 3-37.

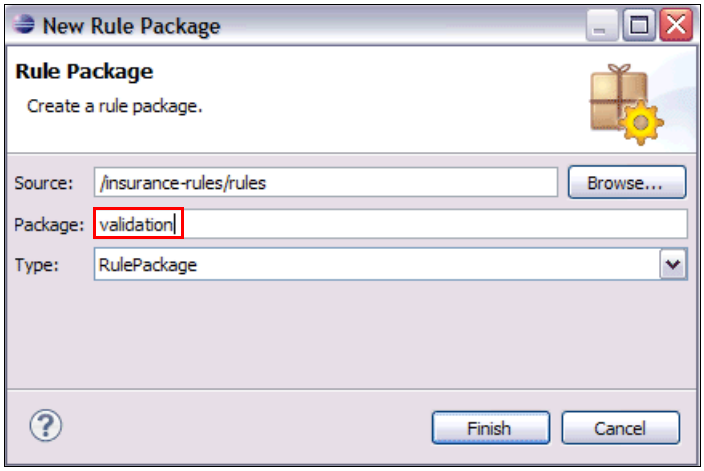


Figure 3-37 Type validation package name

3. To create the ruleflow, in the Orchestrate part of the Rule Project Map, click **Add ruleflow**, as shown in Figure 3-38.

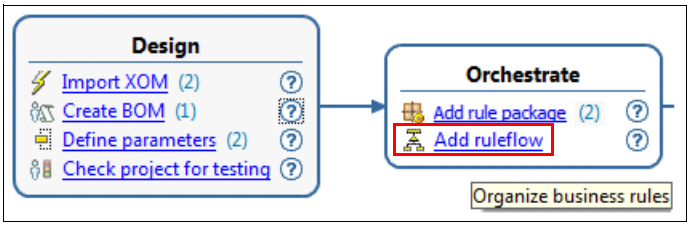


Figure 3-38 Add new ruleflow

4. In the New Ruleflow dialog box, in the Name field, type `mainflow` and then click **Finish**, as shown in Figure 3-39.

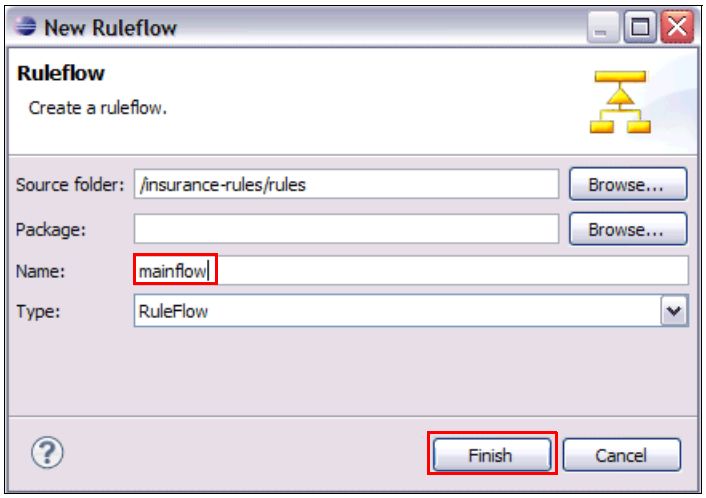


Figure 3-39 Type the name mainflow

5. In the Ruleflow diagram, which is shown in Figure 3-40, complete the following steps:
   a. Add the start node, which is the starting point of the ruleflow. Click the start node icon ( ) in the ruleflow diagram tool bar, and drop it in ruleflow diagram.
   b. Add the end node, which is the endpoint of the ruleflow. Click the end node icon ( ) in ruleflow diagram tool bar, and drop it in the ruleflow diagram.
   c. Add the task for the validation rule package, which is the rule task of the ruleflow. Click the **validation** rule package in the Rule Explorer view, and drag it into the ruleflow diagram.
   d. Now, click the arrow icon ( ), click the start node icon, and then click the **validation** task box. Click the **validation** task box again, and finally click the end node icon.

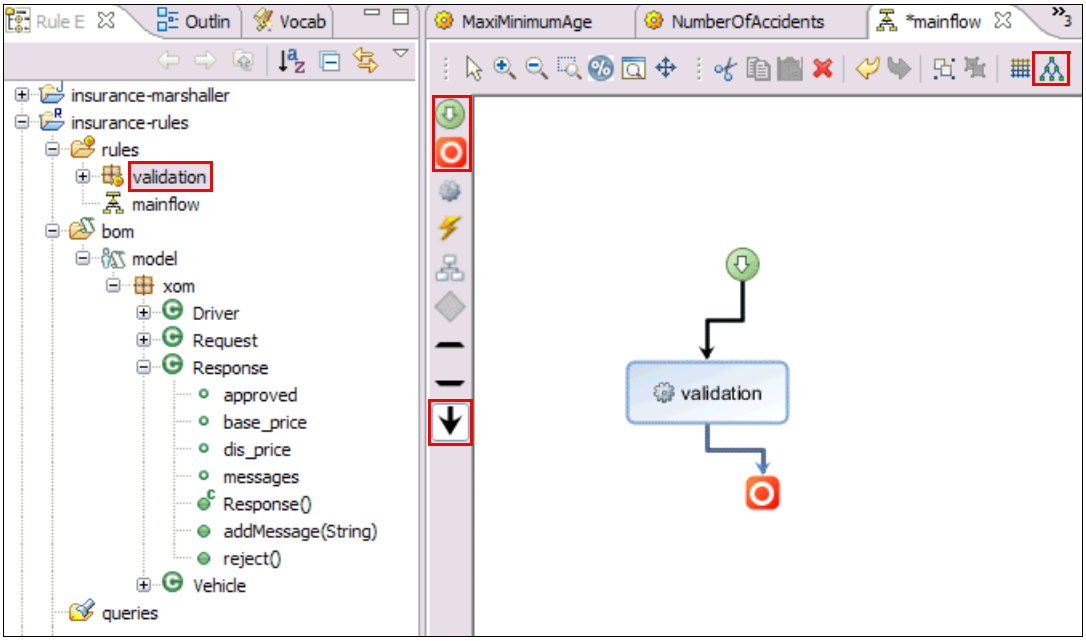


Figure 3-40 Design ruleflow

6. Next, refine the diagram by clicking the button. Figure 3-41 shows the diagram.

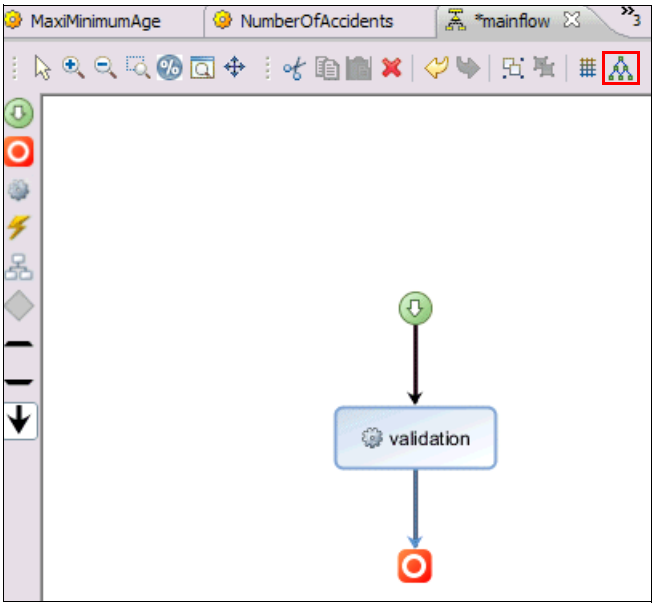


Figure 3-41 Refine the ruleflow

7. Save your work.

3.2.8 Authoring rules

This section explains how to write action rules and put them into the relevant package. You can create the following rules in Rule Designer for the validation packages:

- MaxiMinimumAge rule
- NumberOfAccidents rule

To create the action rules, follow these steps:

1. Create the MaxiMinimumAge rule:

   a. In the rules project, right-click **validation**, and then click **New** → **Action Rule**, as indicated in Figure 3-42.

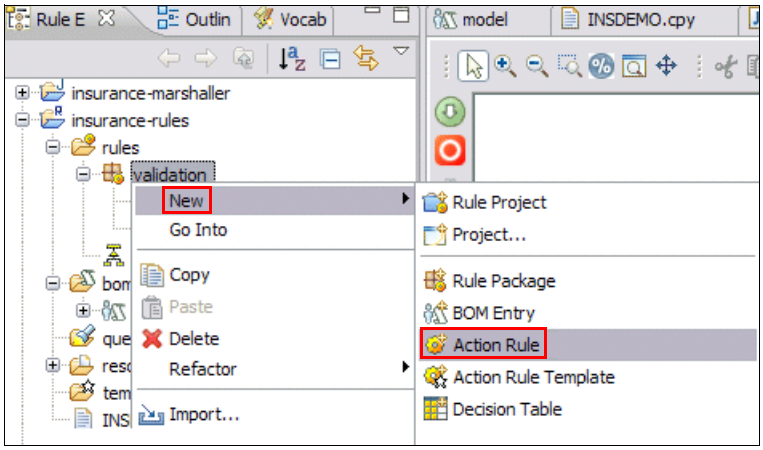


Figure 3-42 Create an action rule

b. In the Name field, enter `MaxiMinimumAge`, as shown in Figure 3-43. Then, click **Finish**.

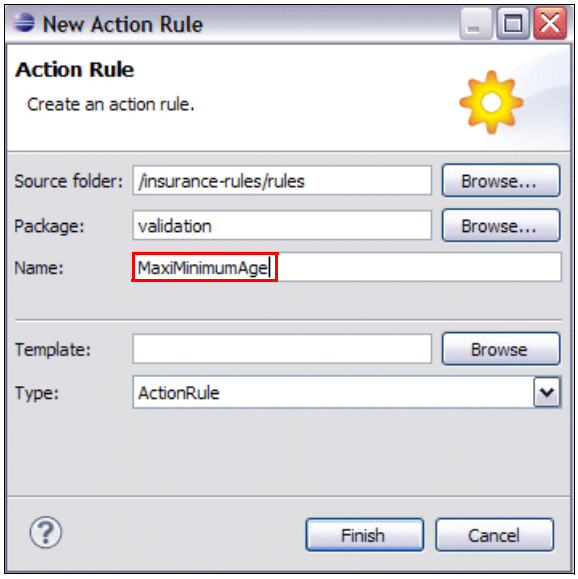


Figure 3-43 Enter the rule name

c. The new action rule displays in the Rule Explorer view, and the Intellirule Editor opens.

d. Type the validation with the MaxiMinimumAge rule, as shown in Example 3-2.

Example 3-2 Validation with the MaxiMinimumAge rule

```
if
    the age of the driver of 'the insurance request' is less than 18
    or the age of the driver of 'the insurance request' is more than 60
then
    add "The age exceeds the maximum or minimum" to the messages of 'the
insurance response' ;
    reject 'the insurance response' ;
```

Figure 3-44 shows the generated action rule.

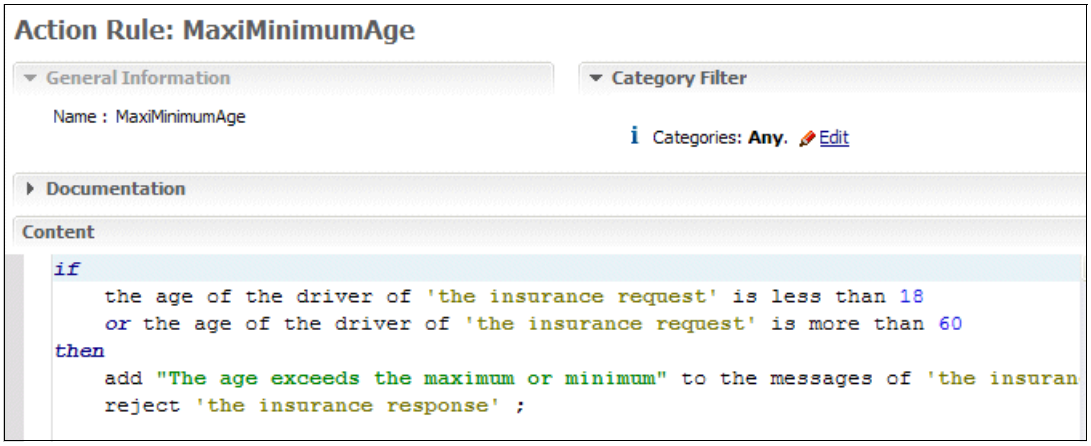


Figure 3-44 View MaxiMinimumAge rule

2. Repeat all the substeps in step 1, and type the validation with the NumberOfAccidents rule, as shown in Example 3-3.

Example 3-3 Validation with the NumberOfAccidents rule

```
if
    the number accidents of the driver of 'the insurance request' is more than
3
then
    add "Accidents number exceeds the maximum" to the messages of 'the
insurance response' ;
    reject 'the insurance response' ;
```

Figure 3-45 shows the generated action rule.

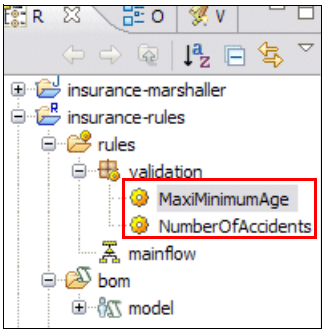


Figure 3-45 View the generated rules

3. Save your work.

Decision Table or Tree: You can also use a Decision Table or Decision Tree to write decision rules, as shown in the example in Figure 3-46.

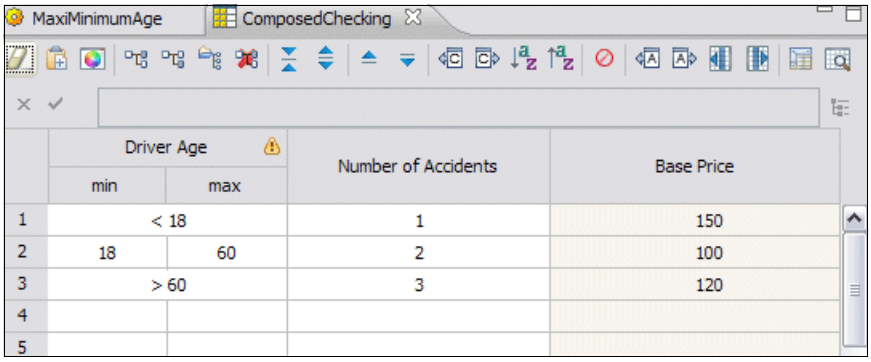


Figure 3-46 Decision table sample

3.2.9 Preparing the rule execution

This section shows how to deploy rules to zRule Execution Server for z/OS and how to view the deployed ruleset on the zRule Execution Server server web console.

Step 1: Create a RuleApp project

First, you must create a RuleApp project to contain the rulesets that you want to execute. To create a RuleApp project, follow these steps:

1. In Rule Designer, in the Deploy and Integrate section of the Rule Project Map, click **Create RuleApp project**, as shown in Figure 3-47.

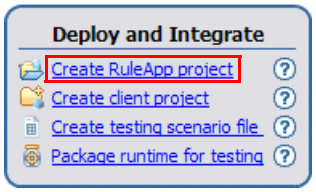


Figure 3-47 Create RuleApp project

2. In the New RuleApp Project wizard, in the Project name field, enter `insuranceApp` as the name for your RuleApp project, as indicated in Figure 3-48.

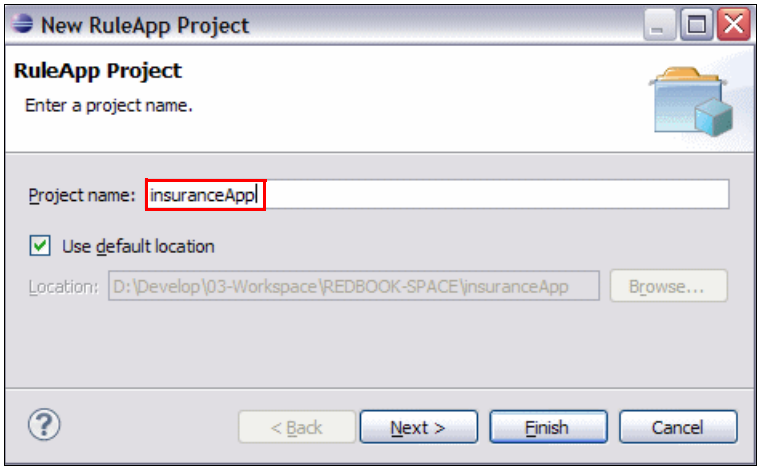


Figure 3-48 Enter RuleApp name

3. Ensure that the “Use default location” option is selected, and click **Next**. The rule project is listed in the Rule Projects tab, as shown in Figure 3-49.

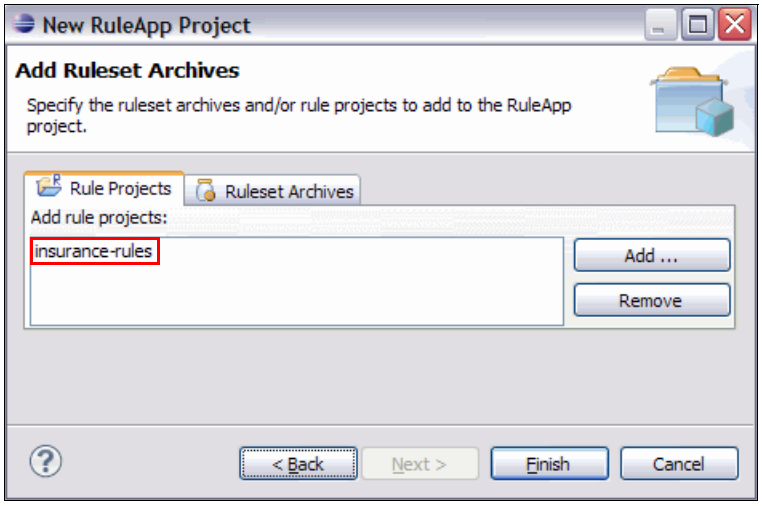


Figure 3-49 View Rule Projects tab

4. Click **Finish**. The RuleApp project is created and displayed in the Rule Explorer view, as shown in Figure 3-50.

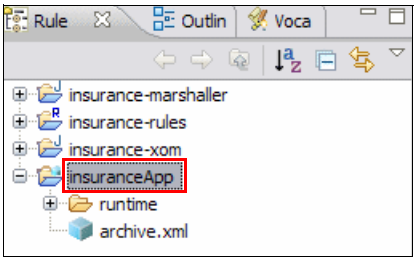


Figure 3-50 The insuranceApp project

Step 2: Deploy the RuleApp to zRule Execution Server for z/OS

To be able to execute the ruleset with zRule Execution Server for z/OS, you must deploy the Java XOM, the marshaller XOM, and your RuleApp to zRule Execution Server for z/OS.

> **Important:** The RuleApp must be deployed to zRule Execution Server for z/OS. Ensure that you start zRule Execution Server for z/OS successfully before you continue this step.

To deploy the XOM, marshaller, and RuleApp, follow these steps:

1. Double-click the **insuranceApp** RuleApp project to open the RuleApp editor.
2. Then, in the Deployment pane, click **Deploy** to deploy the RuleApp to Rule Execution Server, as shown in Figure 3-51.

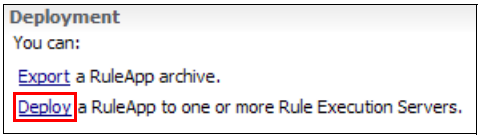


Figure 3-51 Deploy the RuleApp

3. For the deployment type, accept the default option of “Increment RuleApp major version” as shown in Figure 3-52 on page 56, and then click **Next**.

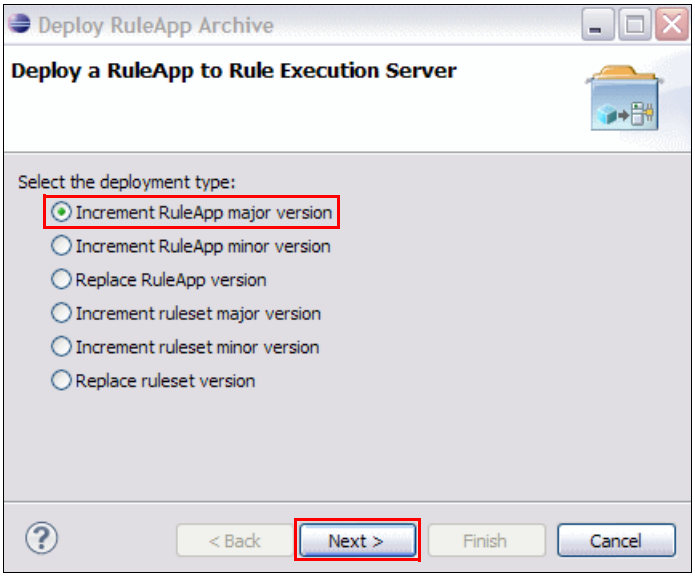


Figure 3-52 Select deployment type

4. Select **Create a temporary Rule Execution Server configuration** and enter the following details, as indicated in Figure 3-53:

   – URL: `http://<your.server.address>:<PORT>/res`
   – Login: `resAdmin`
   – Password: `resAdmin`

   Select **Deploy XOM of rule projects and archives contained in the RuleApp** and click **Finish**.

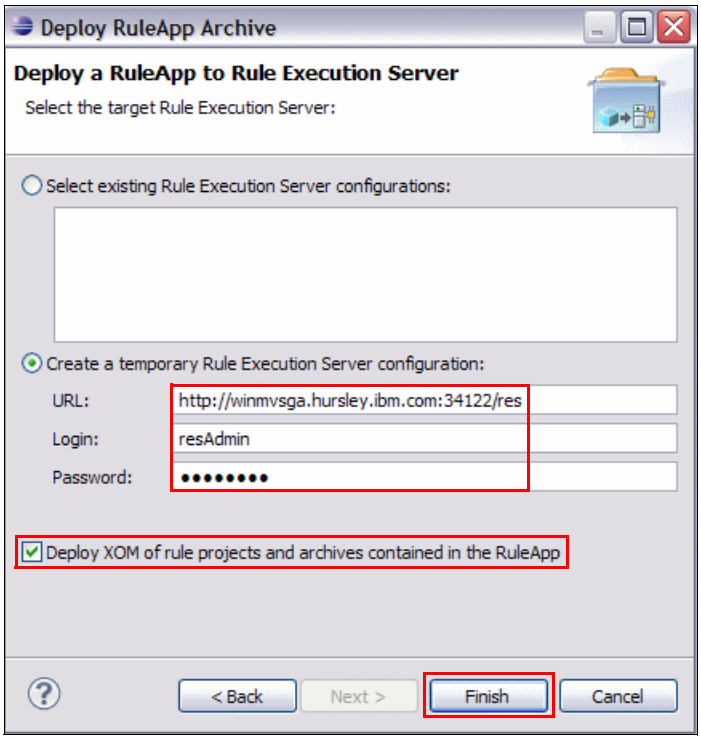


Figure 3-53 Configure the deployment

The artifacts are deployed to zRule Execution Server for z/OS.

Step 3: View deployed rule artifacts in the Rule Execution Server Console

You can log in to the zRule Execution Server Console and use the Navigator pane to view the deployed RuleApp and XOM.

To view your deployed artifacts, follow these steps:

1. In a web browser, open the web console for zRule Execution Server for z/OS using the following URL:

   `http://<your.server.address>:<PORT>/res`

2. At the login prompt, enter the following login details, as indicated in Figure 3-54:
   - Login: `resAdmin`
   - Password: `resAdmin`

Figure 3-54 Log in to Rule Execution Server Console

3. After you log in, click **Explorer**, as indicated in Figure 3-55 on page 58.

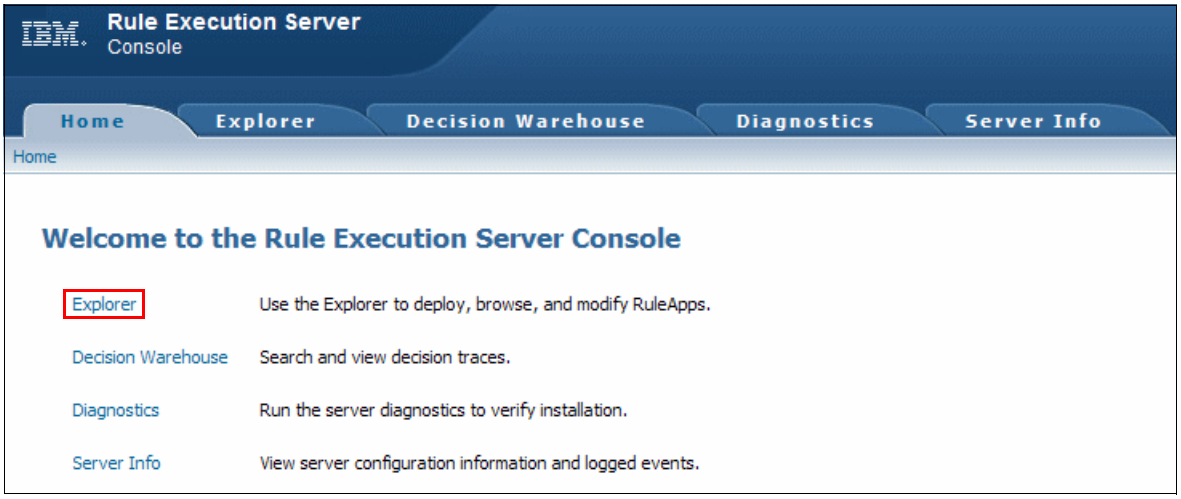


Figure 3-55 Explore the rule project

4. In the Navigator pane, click **RuleApps** to view the deployed RuleApp, as shown in Figure 3-56.

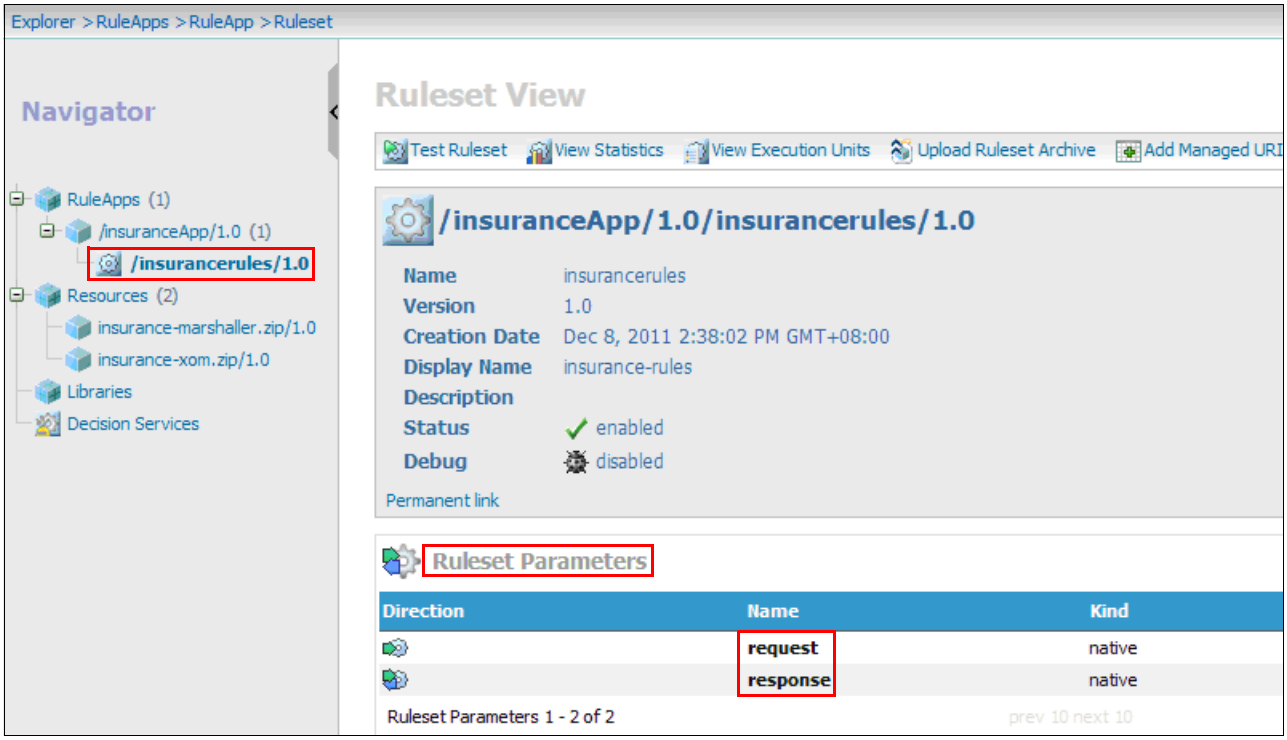


Figure 3-56 View deployed ruleset

5. In the Navigator pane, click **Resources** to view the deployed XOM and the marshaller file, as indicated in Figure 3-57.

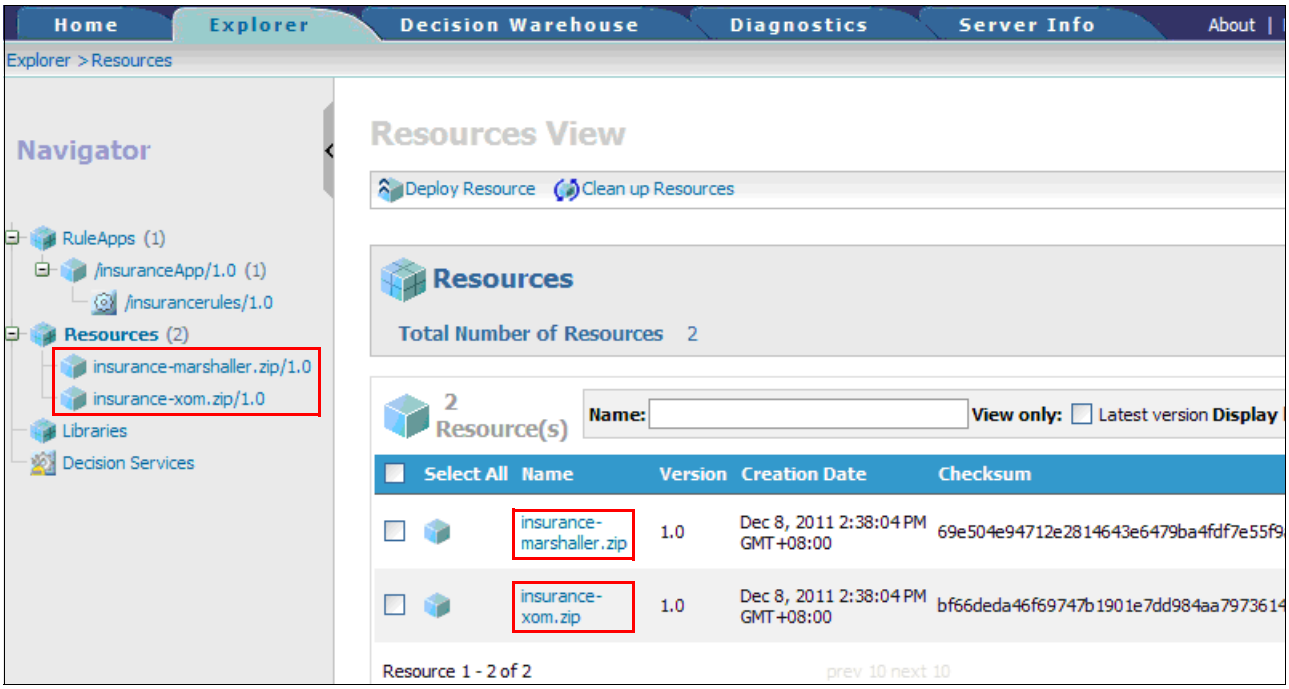


Figure 3-57 View deployed Java XOM and marshaller XOM

3.2.10 Building a COBOL application for rule execution

To execute rules, you call the ruleset from the COBOL application. You can use the zRule COBOL stub API to invoke the rule execution in a running instance of zRule Execution Server for z/OS.

To build the COBOL application, follow these steps:

1. Include the required copybooks for the zRule COBOL stub API:

```
01 WS-REASON-CODES.
COPY HBRC.
COPY HBRWS.
```

2. Specify the ruleset path to initialize the values that are passed to zRule Execution Server for z/OS:

```
* ruleset path from the zRules Execution Server
  MOVE "/insuranceApp/insurancerules"  TO HBRA-CONN-RULEAPP-PATH
```

3. Configure the ruleset parameter:
   - Set the name of the parameter:

     ```
     MOVE 'request' TO HBRA-RA-PARAMETER-NAME(1)
     ```

   - Set the length of the parameter:

     ```
     MOVE LENGTH OF REQUEST TO HBRA-RA-DATA-LENGTH(1)
     ```

   - Set the address of the parameter:

     ```
     SET HBRA-RA-DATA-ADDRESS(1) TO ADDRESS OF REQUEST
     ```

4. Connect to zRule Execution Server for z/OS:

   ```
   CALL 'HBRCONN' USING HBRA-CONN-AREA.
   ```

5. Execute the ruleset:

   ```
   CALL 'HBRCONN' USING HBRA-CONN-AREA.
   ```

6. Disconnect from zRule Execution Server for z/OS:

   ```
   CALL 'HBRDISC' USING HBRA-CONN-AREA.
   ```

COBOL application sample

Example 3-4 includes a sample COBOL application that you can use to call the rules that you designed in the insurance-rules project.

Example 3-4 COBOL application sample to call the rules on zRule Execution Server for z/OS

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. "INSMAIN".
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
           COPY INSDEMO.
       01 WS-REASON-CODES.
           COPY HBRC.
           COPY HBRWS.
       01 WS-MESSAGE-IDX      PIC 9(2).
       01 WS-MAX-TABLE-LEN    PIC 9(18).

       PROCEDURE DIVISION.
      * Init ruleset parameter data
           MOVE 'John'             TO FIRST-NAME
           MOVE 'Smith'            TO LAST-NAME
           MOVE 'XA123456'         TO ZIPCODE
           MOVE 123456             TO HOUSE-NUM
           MOVE 17                 TO AGE
           MOVE '20110908'         TO LIC-DATE
           MOVE 'F'                TO LIC-STATUS
           MOVE 4                  TO NUMBER-ACCIDENTS
           MOVE 'F'                TO APPROVED
           MOVE 100                TO BASE-PRICE
           MOVE 0                  TO MSG-COUNT
      * Move ruleset parameters to table HBRA-RA-PARMETERS
           MOVE ZERO               TO HBRA-CONN-RETURN-CODES
           MOVE LOW-VALUES         TO HBRA-RA-PARMETERS
           MOVE "/insuranceApp/insurancerules"
                                   TO HBRA-CONN-RULEAPP-PATH
      * Parameter Borrower
           MOVE LOW-VALUES         TO HBRA-RA-PARMETERS.
           MOVE 'request'          TO HBRA-RA-PARAMETER-NAME(1)
           MOVE LENGTH OF REQUEST  TO HBRA-RA-DATA-LENGTH(1)
           SET HBRA-RA-DATA-ADDRESS(1)
                                   TO ADDRESS OF REQUEST
      * Parameter Loan
           MOVE 'response'         TO HBRA-RA-PARAMETER-NAME(2)
           MOVE LENGTH OF RESPONSE TO HBRA-RA-DATA-LENGTH(2)
      * For ODO Table, the length represent the max length.
```

```
        COMPUTE WS-MAX-TABLE-LEN = LENGTH OF Messages * 100
        ADD WS-MAX-TABLE-LEN    TO HBRA-RA-DATA-LENGTH(2)
        SET HBRA-RA-DATA-ADDRESS(2)
                                TO ADDRESS OF RESPONSE
    * Get connection to rule execution server
        CALL 'HBRCONN' USING HBRA-CONN-AREA.
        IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK
            DISPLAY "connect zRules failed"
            DISPLAY "CC code " HBRA-CONN-COMPLETION-CODE
            DISPLAY "RC code " HBRA-CONN-REASON-CODE
            DISPLAY "Message " HBRA-RESPONSE-MESSAGE
        ELSE
            DISPLAY 'connect zRules successful'
        END-IF
    * Invoke rule execution server
        CALL 'HBRRULE' USING HBRA-CONN-AREA
        IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK
            DISPLAY "invoke zRules failed"
            DISPLAY "CC code " HBRA-CONN-COMPLETION-CODE
            DISPLAY "RC code " HBRA-CONN-REASON-CODE
            DISPLAY "Message " HBRA-RESPONSE-MESSAGE
        ELSE
            DISPLAY 'invoke zRules successful'
        END-IF
    * Get disconnect to rule execution server
        CALL 'HBRDISC' USING HBRA-CONN-AREA
    * Display result
        DISPLAY         "********** EXECUTION RESULT *********"
        DISPLAY         "DRIVER NAME: " FIRST-NAME
        DISPLAY         "RESPONSE APPROVED: " APPROVED
        IF approved = "F"
           DISPLAY      "Reject messages:"
           PERFORM VARYING WS-MESSAGE-IDX FROM 1 BY 1
                           UNTIL WS-MESSAGE-IDX > MSG-COUNT
              DISPLAY messages (WS-MESSAGE-IDX)
           END-PERFORM
        END-IF
        DISPLAY         "***************************************"
    STOP RUN.
```

Additional resources: You can find the `INSMAIN.cbl` application sample in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. Refer to Appendix A, “Additional material” on page 259.

Rule execution

You can compile and run the COBOL application on z/OS. In the COBOL application sample that is shown in Example 3-4 on page 60, you hard-coded the following input values:

- AGE: `17`
- NUMBER-ACCIDENTS: `4`

Example 3-5 shows the results after the rule execution.

Example 3-5 Rule execution result

```
********** EXECUTION RESULT *********
DRIVER NAME: John
RESPONSE APPROVED: F
Reject messages:
Accidents number exceeds the maximum
The age exceeds the maximum or minimum
*************************************
```

3.3 Getting started from an existing rule project

This section provides guidance about how to share business rules from an existing Java-based rule project to a COBOL application on z/OS.

3.3.1 Overview

In this scenario, an insurance company has an existing business rule application to perform user validation for an insurance application. The rule projects, which the company currently uses, contain BOM that is based on a Java XOM. The company deploys the rules to the Rule Execution Server in a distributed environment.

The company now wants to share the Java rule projects with COBOL applications that run on z/OS and to manage the changes that are made to these rules. To share rules with COBOL applications, the company must add the necessary COBOL structures to the BOM and then generate a COBOL copybook. With these structures in the rule project, the company can then deploy the rules application to zRule Execution Server for z/OS so that the COBOL application can call the rulesets and execute the rules.

This section provides an existing rule project, sharinginsurance-rules, that has a BOM that is generated from a Java XOM (sharinginsurance-xom). It also uses a RuleApp project (sharinginsuranceApp) that is used for rule deployment to the runtime environment.

You can import the existing rule project from the source code that is delivered with this book. Refer to Appendix A, “Additional material” on page 259.

Figure 3-58 shows the existing rule project structure in Rule Designer.

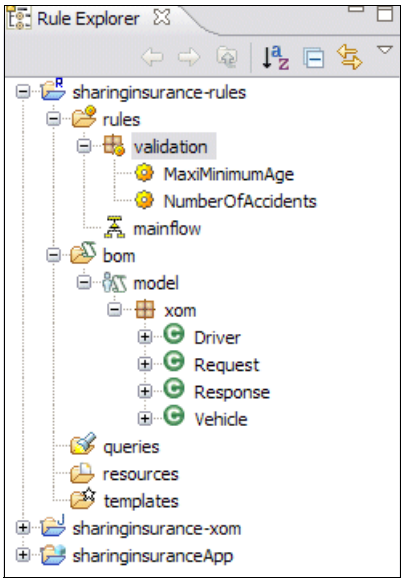


Figure 3-58 Existing rule project structure

3.3.2 Generating a copybook from the BOM

You use the COBOL Enabled BOM feature of Rule Designer to generate a copybook from the BOM in the existing rule project.

Use the following configuration for the BOM:

- Specify each Java class type that you want to use as top-level data items in the copybook.
- Enter a name for the runtime marshaller project and package, which are created during the copybook generation.

To configure the BOM for copybook generation, follow these steps:

1. In the Rule Explorer, right-click the **sharinginsurance-rules** rule project, and then select **Properties** → **COBOL Management** → **COBOL Enabled BOM**. Then, click **Add**, as indicated in Figure 3-59.

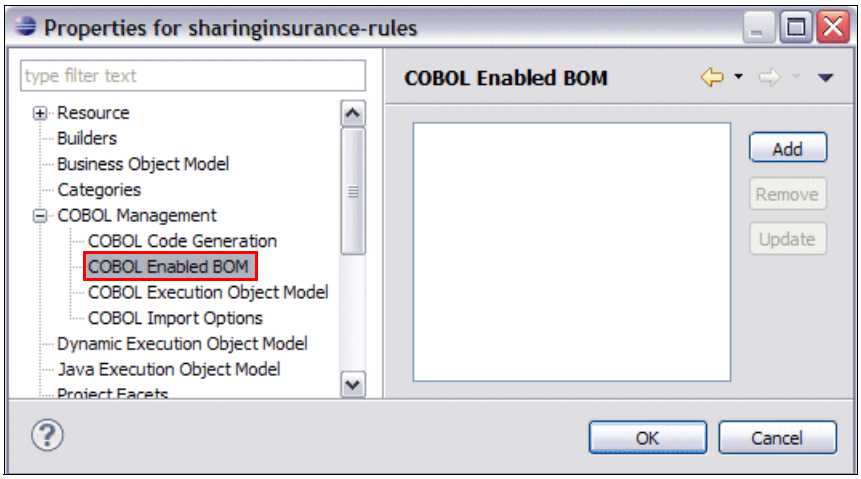


Figure 3-59 Navigate to COBOL Enabled BOM

2. In the Enable COBOL support for BOM entry window, click **Browse** to select the BOM entry that you want to enable for COBOL, as shown in Figure 3-60.

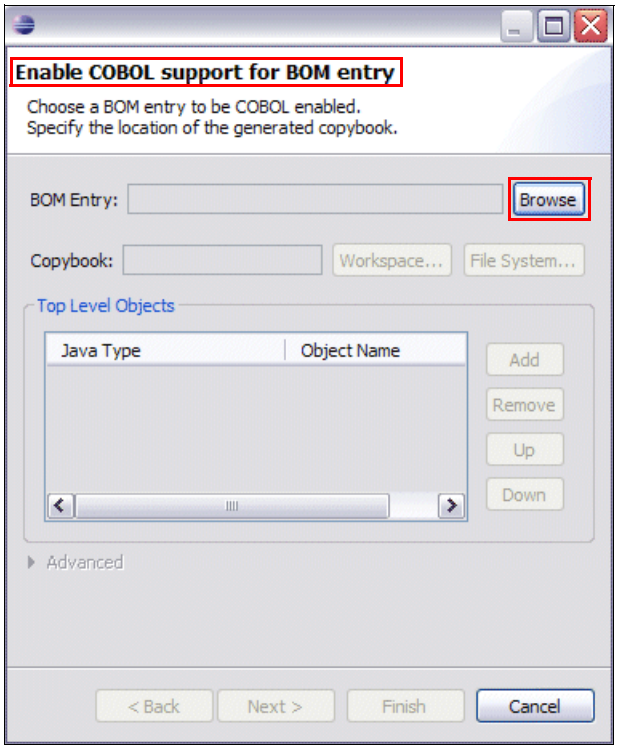


Figure 3-60 Go to Enable COBOL wizard

3. In the Select BOM entry dialog box, select **model**, and then click **OK**, as indicated in Figure 3-61.

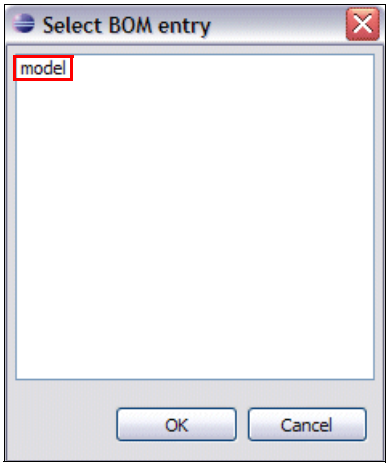


Figure 3-61 Select BOM model

4. Back on the Enable COBOL support for BOM entry dialog box, click **Workspace**, and specify the path of the copybook to be generated. In this case, specify `/sharinginsurance-rules/INSSHAR.cpy`, as shown in Figure 3-62. Click **OK**.

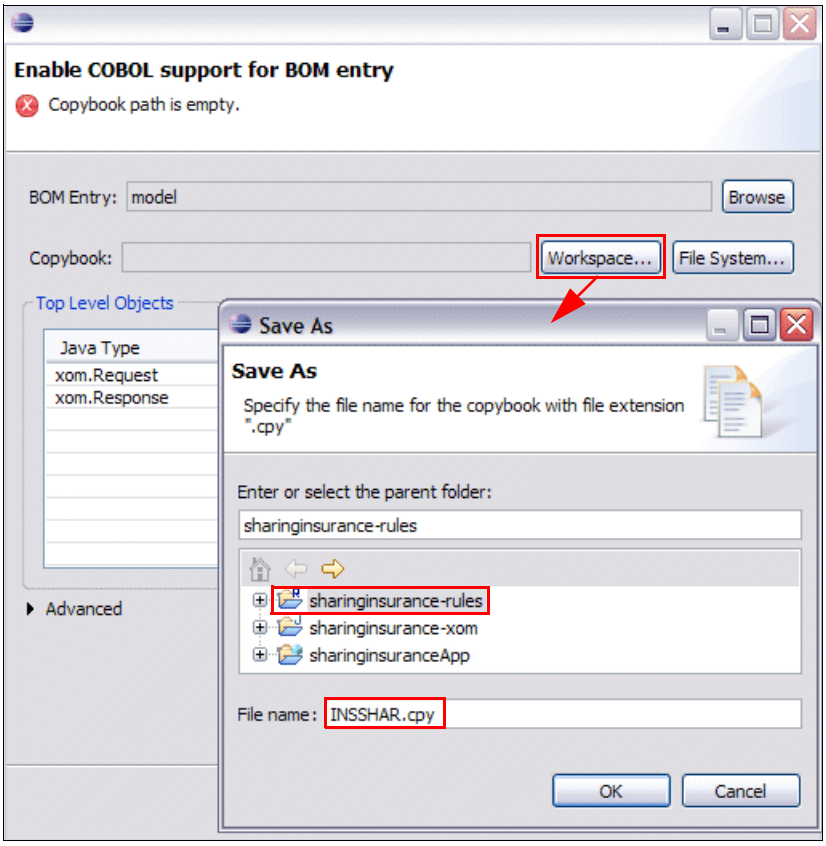


Figure 3-62 Specify the path of the copybook to generate

5. In the Top Level Objects section, accept xom.Request and xom.Response as the top-level data items, as indicated in Figure 3-63 on page 66, and click **Next**.

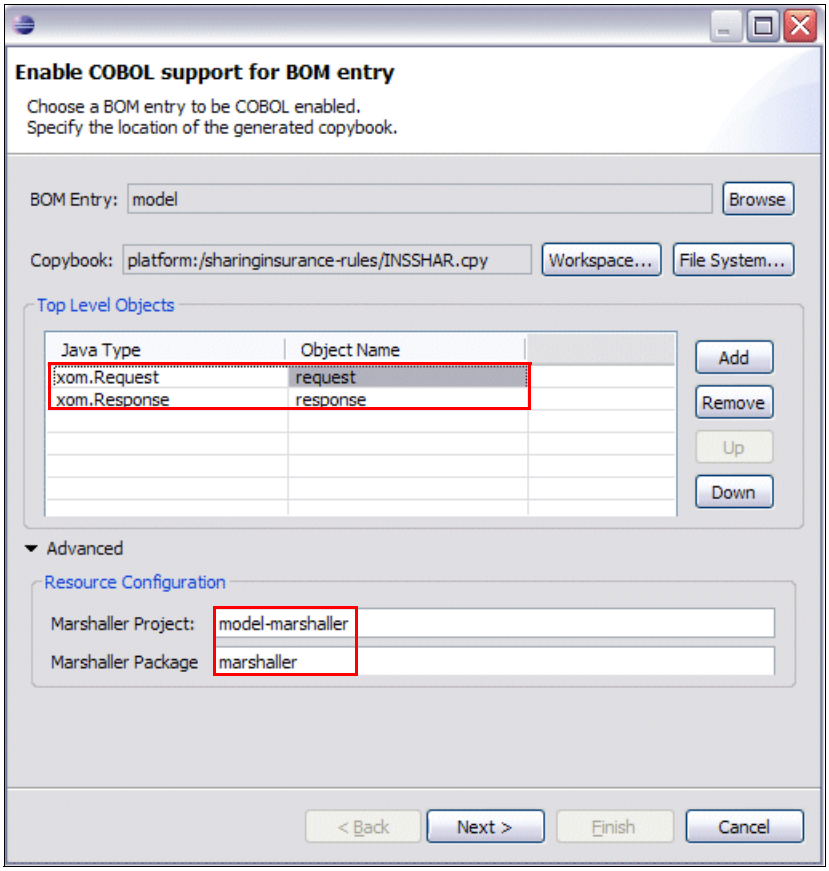


Figure 3-63 Configure top-level objects and marshaller project

6. Under Object Name, accept the request and response names that correspond to the Java types. When you generate the COBOL copybook, these names are given to the top-level data items.
7. In the Resource Configuration section, accept the default names for the runtime Marshaller Project and Marshaller Package. Click **Next**.

   A table shows the proposed mapping between the Java structures in the BOM and the COBOL structures, as shown in Figure 3-64 on page 67.

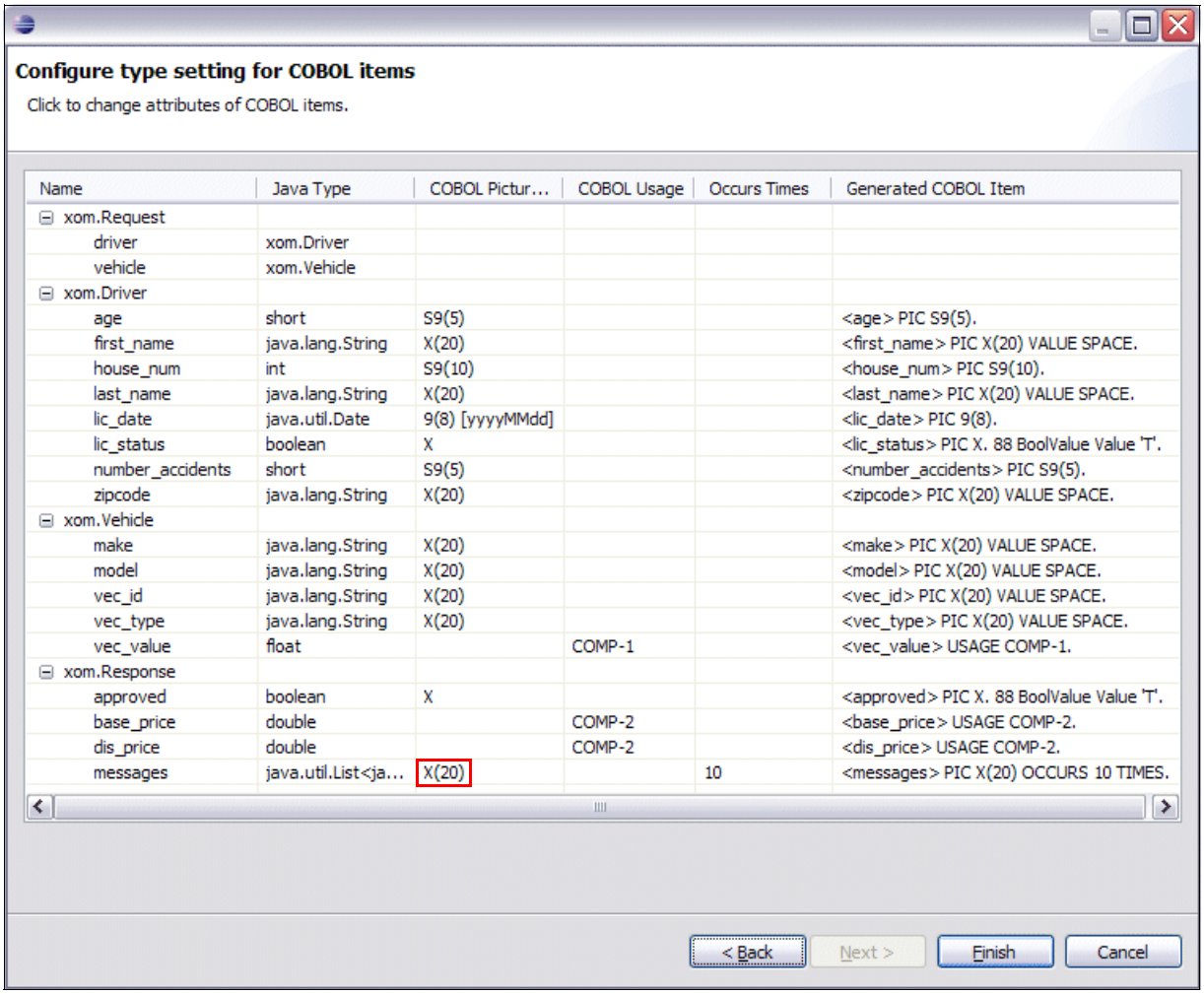


Figure 3-64 Configure BOM to COBOL type mapping

8. Click the COBOL Picture field for the messages item of the xom.Response class in the Configure type setting for COBOL items window, and change the default length for messages from X(20) to X(60). See Figure 3-65.

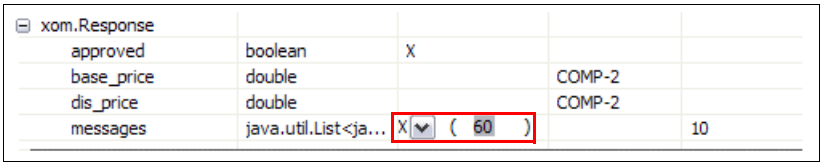


Figure 3-65 Change the default mapping

> **Important:** The default mapping from Java String to COBOL Picture length is 20. You adjust this value per rule project. In this scenario, the sharinginsurance-rules project uses a COBOL Picture length mapping value of 60, as required by the real reject message in rules.

9. Click **Finish**.

   The BOM model is now listed as a COBOL-enabled BOM, as shown in Figure 3-66.

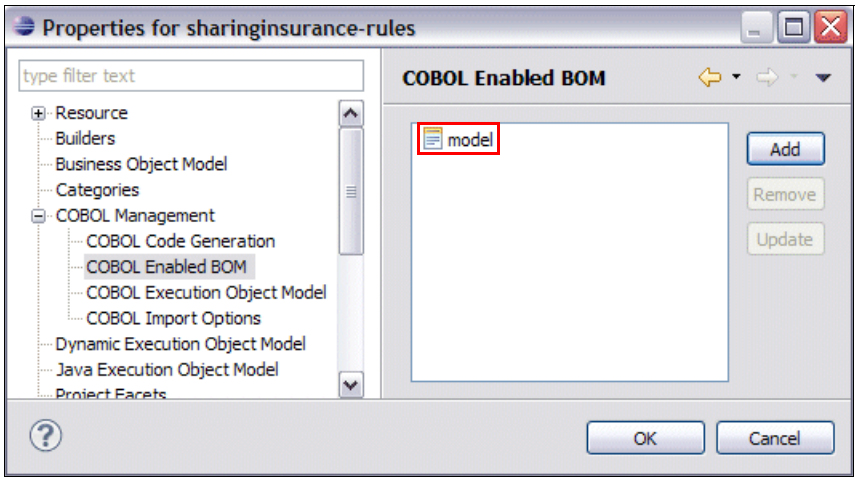


Figure 3-66 COBOL Enabled BOM

10. Click **OK**.

You can view the following generated copybook and marshaller project, as shown in Figure 3-67:

- `INSSHAR.cpy` generated copybook
- `model-marshaller` generated marshaller project

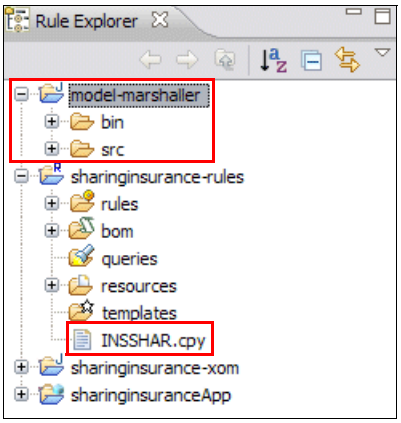


Figure 3-67 Generated copybook and marshaller

Important: Do not change the generated copybook or marshaller project. When a change occurs to the BOM of the rule project, use the COBOL Enabled BOM feature to update the copybook and marshaller project.

3.3.3 Deploy rule artifacts to zRule Execution Server for z/OS

To execute a ruleset with zRule Execution Server for z/OS, you must deploy the rule project and the Java XOM with the marshaller XOM to zRule Execution Server for z/OS. The deployment process is same as the process that is described in 3.2, “Getting started from a COBOL copybook” on page 29.

Important: Ensure that you started zRule Execution Server for z/OS successfully before you attempt to deploy rules.

To deploy rule artifacts to zRule Execution Server for z/OS, follow these steps:

1. Double-click the **sharinginsuranceApp** project to open the RuleApp editor. Then, in the Deployment pane, click **Deploy** to deploy the RuleApp to zRule Execution Server for z/OS.
2. For the deployment type, accept the “Increment RuleApp major version” default option, and then click **Next**.
3. Select the “Create a temporary Rule Execution Server configuration” option, and enter the following details:
   - URL: `http://<your.server.address>:<PORT>/res`
   - Login: `resAdmin`
   - Password: `resAdmin`
4. Click **Finish**.

Your artifacts are deployed to zRule Execution Server for z/OS. You can build a COBOL application to invoke the rule execution.

3.3.4 Building a COBOL application for rule execution

You deployed the rule artifacts to zRule Execution Server for z/OS. You can use the generated copybook to build a COBOL application for rule execution on z/OS, as described in the following sections.

Generated copybook sample

First, you must know the structure of the generated copybook, as shown in Example 3-6.

Example 3-6 Generated copybook INSSHAR.cpy

```
01 request.
           02 driver.
               03 age pic S9(5).
               03 first-name pic X(20) value SPACE.
               03 house-num pic S9(10).
               03 last-name pic X(20) value SPACE.
               03 lic-date pic 9(8).
               03 lic-status pic X.
                   88 BoolValue value 'T'.
               03 number-accidents pic S9(5).
               03 zipcode pic X(20) value SPACE.
           02 vehicle.
               03 make pic X(20) value SPACE.
               03 model pic X(20) value SPACE.
               03 vec-id pic X(20) value SPACE.
               03 vec-type pic X(20) value SPACE.
               03 vec-value usage COMP-1.
       01 response.
           02 approved pic X.
               88 BoolValue value 'T'.
           02 base-price usage COMP-2.
           02 dis-price usage COMP-2.
           02 messages-Num pic 9(9).
           02 messages pic X(60) value SPACE Occurs 10 Times.
```

COBOL application sample

Now, you can build a COBOL application according to the generated copybook, as shown in Example 3-7.

Example 3-7 COBOL application INSSHAR.cbl

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. "INSSHAR".
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      * include the generated copybook
           COPY INSSHAR.
       01 WS-REASON-CODES.
           COPY HBRC.
           COPY HBRWS.
       PROCEDURE DIVISION.
      * Init ruleset parameter data
           MOVE 'John'             TO FIRST-NAME
           MOVE 17                 TO AGE
           MOVE 4                  TO NUMBER-ACCIDENTS
       ......
      * Move ruleset path to table HBRA-RA-PARMETERS
      ......
           MOVE "/sharinginsuranceApp/sharinginsurancerules"
                                   TO HBRA-CONN-RULEAPP-PATH
      * move ruleset parameter for request and response
           MOVE 'request'          TO HBRA-RA-PARAMETER-NAME(1)
      ......
           MOVE 'response'         TO HBRA-RA-PARAMETER-NAME(2)
        ......
      * Get connection to rule execution server
           CALL 'HBRCONN' USING HBRA-CONN-AREA.
      ......
      * Invoke rule execution server
           CALL 'HBRRULE' USING HBRA-CONN-AREA
      ......
      * Get disconnect to rule execution server
           CALL 'HBRDISC' USING HBRA-CONN-AREA
      ......
      * Display result
           DISPLAY         "RESPONSE APPROVED: " APPROVED
      ......
     STOP RUN.
```

Additional resources: You can find this sample in the additional information that is included in this book. Refer to Appendix A, “Additional material” on page 259.

Rule execution result

You can compile and run the COBOL application on z/OS. The COBOL application sample in Example 3-7 on page 70 produces the results that are shown in Example 3-8.

Example 3-8 Result of compiling and running the COBOL application

```
********** EXECUTION RESULT *********
DRIVER NAME: John
RESPONSE APPROVED: F
Reject messages:
Accidents number exceeds the maximum
The age exceeds the maximum or minimum
*************************************
```

4

Decision server events

This chapter enhances the request for a quote scenario (introduced in 1.6, “Overview of the scenario used in this book” on page 7) using IBM WebSphere Operational Decision Management for z/OS to perform event pattern detection. It includes the following topics:

- Scenario overview
- Building the event application
- Deploying the event application to the event run time
- Emitting events from CICS
- Running the scenario
- Using connectors to receive events from various z/OS sources

© Copyright IBM Corp. 2012. All rights reserved.

4.1 Scenario overview

In this version of the scenario, the request for quote application running on CICS is designed to emit (create) a business event each time that a quote is requested.

Using this event, WebSphere Operational Decision Management identifies the situation where a customer submits more than three requests for a quotation within the same hour. When this situation is identified, WebSphere Operational Decision Management generates an action that notifies the sales team to follow up with the customer to ensure that the policy is purchased.

Figure 4-1 shows the event processing overview for the request for a quote scenario.

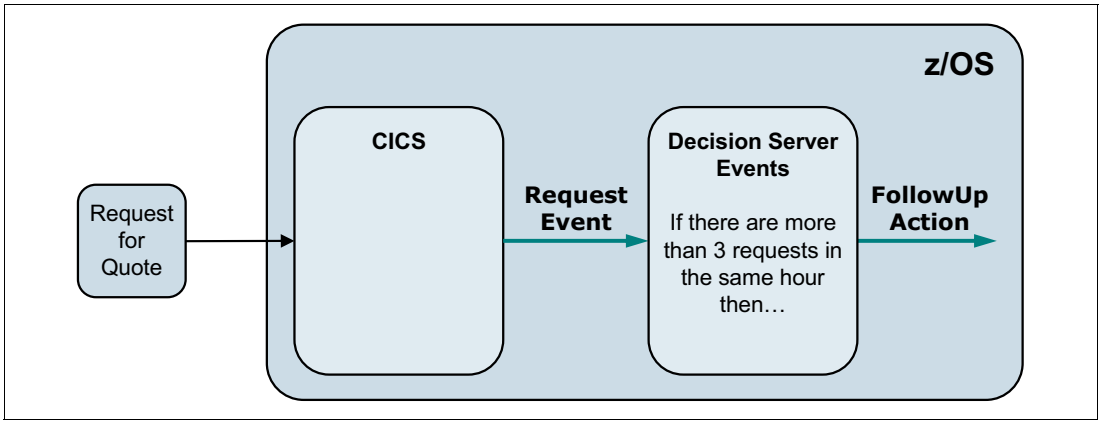


Figure 4-1 The event processing overview for the request for quote scenario

The business event that is generated for each quotation request is modeled on the original COBOL copybook, which is described in Chapter 3, “Getting started with business rules” on page 27 and is included in Example 4-1.

Example 4-1 The COBOL copybook describing a customer’s request

```
01  REQUEST.
    05  DRIVER.
        10  FIRST-NAME                PIC  X(20).
        10  LAST-NAME                 PIC  X(20).
        10  ZIPCODE                   PIC  X(8).
        10  HOUSE-NUM                 PIC  9(8).
        10  AGE                       PIC  9(2) USAGE COMP-3.
        10  LIC-DATE                  PIC  X(8).
        10  LIC-STATUS                PIC  X.
        10  NUMBER-ACCIDENTS    PIC  99.
    05  VEHICLE.
        10  VEC-ID                    PIC  X(15).
        10  MAKE                      PIC  X(20).
        10  MODEL                     PIC  X(20).
        10  VEC-VALUE                 USAGE COMP-1.
        10  VEC-TYPE                  PIC  X(2).
            88 SUV     VALUE 'SU'.
            88 SEDAN   VALUE 'SD'.
            88 PICKUP  VALUE 'PU'.
```

Additional resource: You can find the `INSDEMO.cpy` copybook file in the additional information that is included in this book in the `code/Chapter3/CopybookBased` directory. Refer to Appendix A, “Additional material” on page 259. You can ignore the 01 level RESPONSE structure, because it is required only for rule executions.

The copybook describes the data structure of both the driver and the vehicle in the scenario. The copybook is used in the Event Designer to build an event application. In 4.2, “Building the event application” on page 75, we describe the steps to build the application. In 4.3, “Deploying the event application to the event run time” on page 88, we describe how to deploy the event application to the Decision Server Events run time.

In 4.4, “Emitting events from CICS” on page 92, the CICS Event Binding Editor is used to build an event within a CICS bundle that tells CICS how to emit the business event when a customer requests a quotation. In 4.5, “Running the scenario” on page 104, we then describe how to deploy a CICS COBOL application to trigger the emission of the request for quote event. We use the Decision Server Events tooling to show the events that are received and the action that is fired.

Finally, 4.6, “Using connectors to receive events from various z/OS sources” on page 108 describes how business events can be received from other sources on z/OS, such as files and message queues.

4.2 Building the event application

In this section, we explain how to use the Event Designer to build the event application. The Event Designer is a Decision Server Events component that supports the definition of the metadata layer that is required for business event processing (BEP). You can use the Event Designer to create all the building blocks for your event application, including events, business objects, actions, and event rules. The Event Designer is an Eclipse-based tool that can be run on a Microsoft Windows or Linux workstation.

4.2.1 Event project overview

The Event Designer uses an event project to store all the artifacts that are required for event processing. Figure 4-2 illustrates an overview of the artifacts that can be created in an event project.

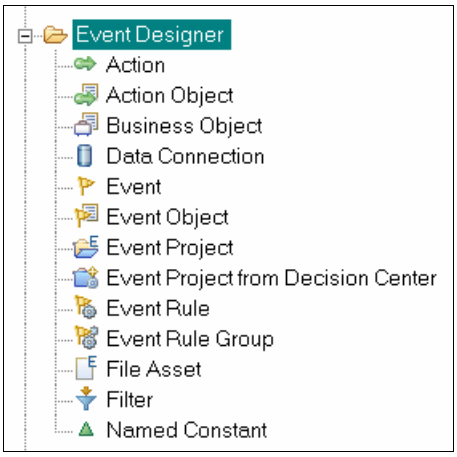


Figure 4-2 Event project artifacts

For more information about the artifacts that are created in an event project, review the event projects topic in the WebSphere Operational Decision Management Information Center:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.reference.doc/doc/webspherebusinesseventsprojects.html

4.2.2 Creating the event project

To create the event project in the Event Designer, follow these steps:

1. First, start the Event Designer in Windows by clicking **Start** → **Programs** → **IBM WebSphere Operational Decision Management V7.5** → **WebSphere Decision Server V7.5** → **Event Designer**.
2. Select **File** → **New** → **Event Project**, as shown in Figure 4-3 on page 77.

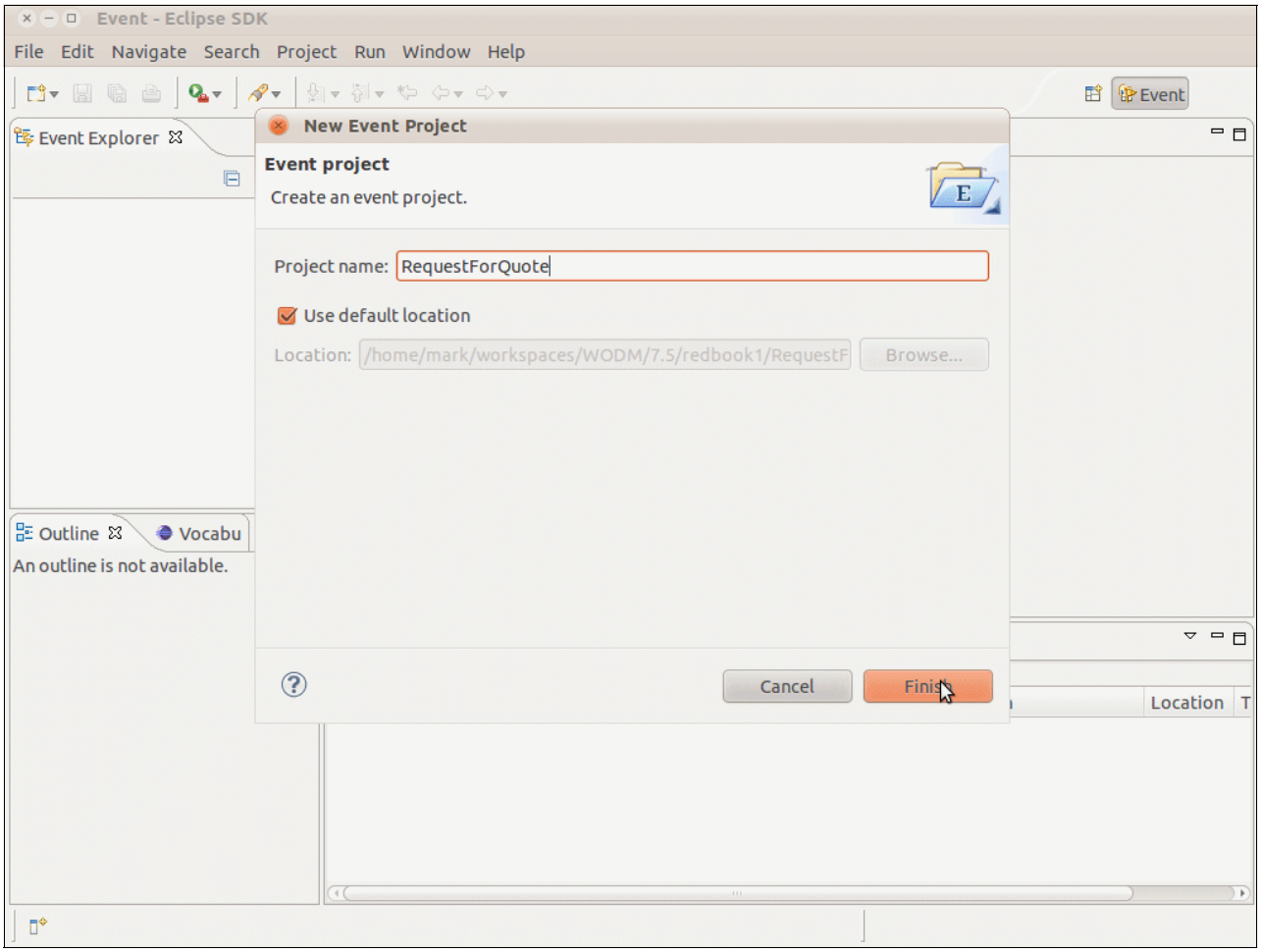


Figure 4-3 Creating the event project

3. Provide the name of the new event project as `RequestForQuote`, and click **Finish**.

4.2.3 Creating the business objects from a COBOL copybook

In the new event project, you create the business objects that represent the vehicle and driver structures in the COBOL copybook. In Rule Designer, described in Chapter 3, “Getting started with business rules” on page 27, the copybook can be imported to automatically generate the required objects. There is no equivalent functionality in the Event Designer, and therefore, you must create the business objects manually.

Creating the business object

To define the first business object, follow these steps:

1. Right-click the new **RequestForQuote** project, and select **New** → **Business Object**.
2. Enter the name `Vehicle`, as shown in Figure 4-4, and click **Next**.

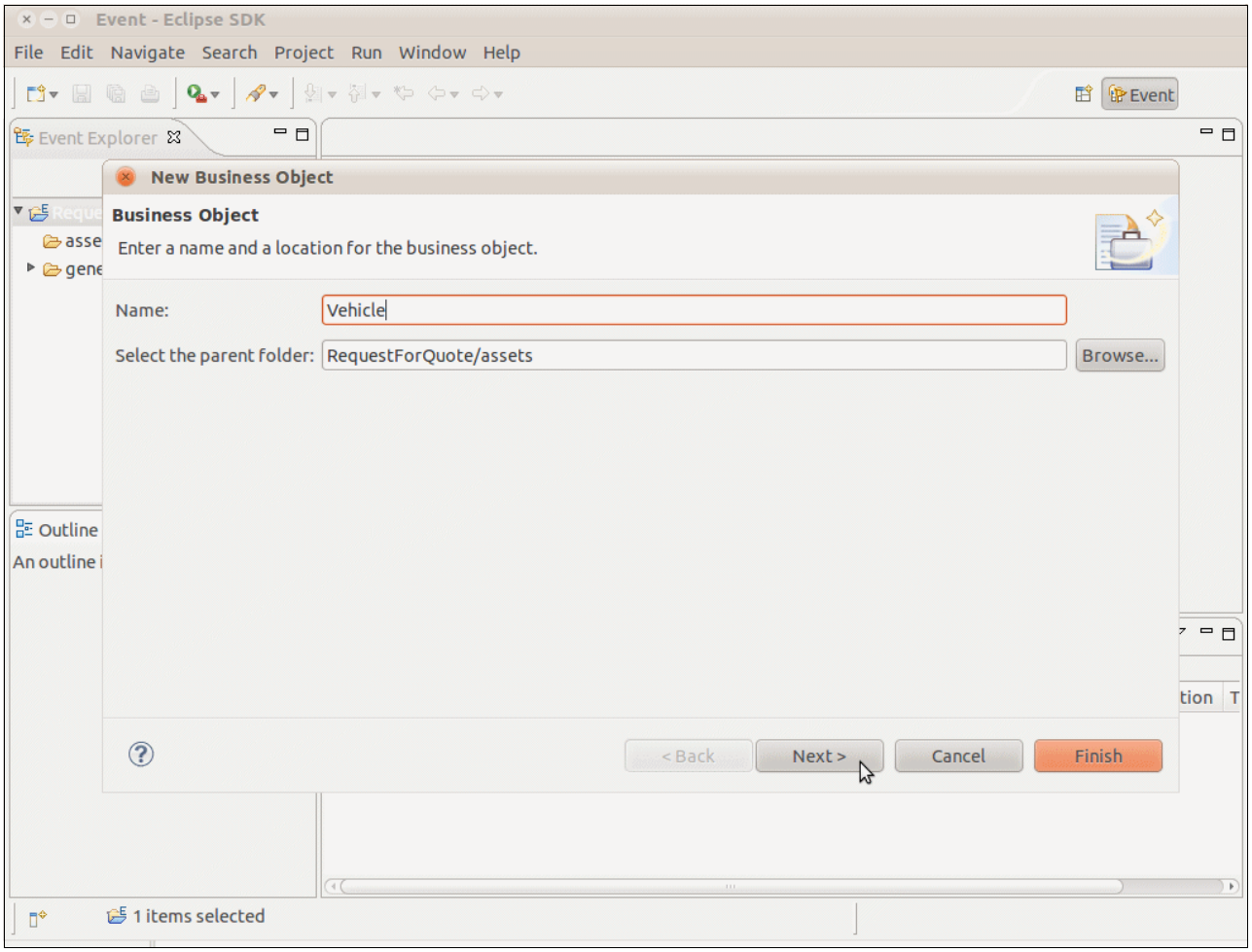


Figure 4-4 Creating the business objects

3. Accept the “Start with a blank business object” default option, and click **Finish**.
4. Repeat this procedure, but enter the name `Driver` to create the driver business object.

Adding fields to the business object

After you create the business objects, you add the necessary fields to these objects. These fields store the information that is required by the event logic at run time.

To add a field to the Vehicle business object, follow these steps:

1. Double-click the **Vehicle** business object to open the edit view. Then, click **Add** in the Field table to add the first field, as shown in Figure 4-5.

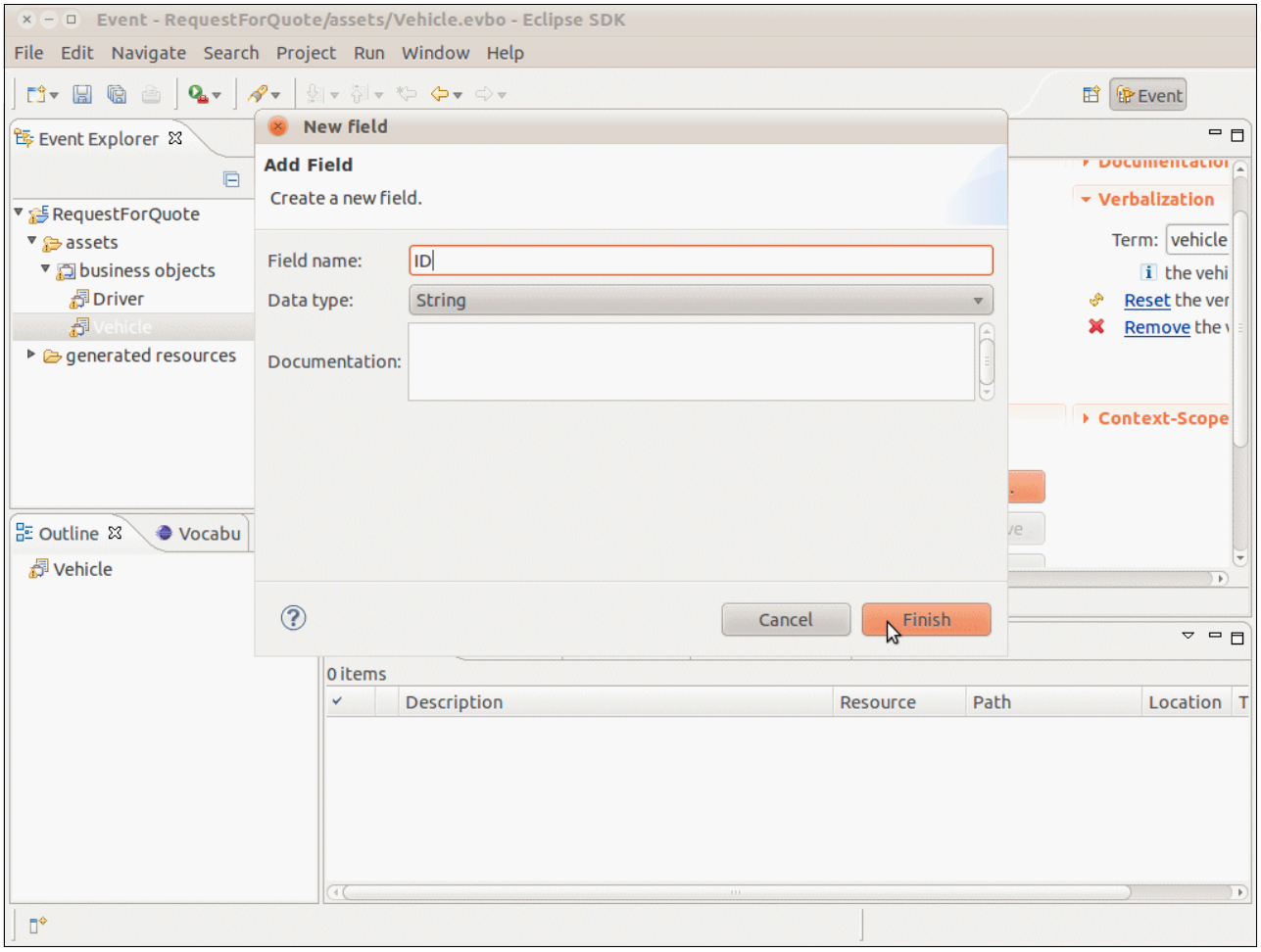


Figure 4-5 Adding fields to the business object

2. Enter the name `ID`, select the **String** data type, and then, click **Finish**.
3. Go to the Field tab at the bottom of the editor, and ensure that the verbalization of the field is set, for example, `{ID} of {this}`. If the verbalization is set, click **Create a default verbalization**.
4. Repeat these steps for all the vehicle fields that are shown in Figure 4-6 on page 80:
   - Enter the name `Make`, select the **String** data type, and then, click **Finish**.
   - Enter the name `Model`, select the **String** data type, and then, click **Finish**.
   - Enter the name `Value`, select the **Integer** data type, and then, click **Finish**.
   - Enter the name `Type`, select the **String** data type, and then, click **Finish**.

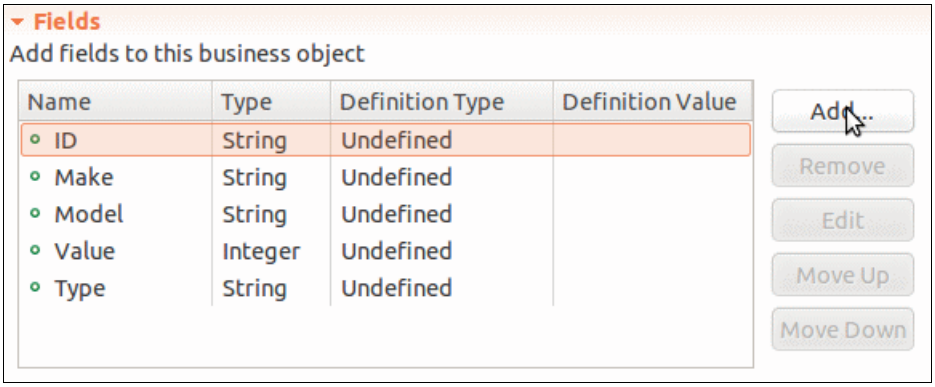


Figure 4-6 All fields required in the Vehicle business object

5. Now open and add fields to the Driver business object by repeating step 1 on page 79 to step 4 on page 79, as shown in Figure 4-7, and add the fields:
   - Enter the name `First Name`, select the **String** data type, and then, click **Finish**.
   - Enter the name `Last Name`, select the **String** data type, and then, click **Finish**.
   - Enter the name `Zip Code`, select the **String** data type, and then, click **Finish**.
   - Enter the name `House Number`, select the **Integer** data type, and then, click **Finish**.
   - Enter the name `Age`, select the **Integer** data type, and then, click **Finish**.
   - Enter the name `License Date`, select the **String** data type, and then, click **Finish**.
   - Enter the name `License Status`, select the **String** data type, and then, click **Finish**.
   - Enter the name `Number Accidents`, select the **Integer** data type, and then, click **Finish**.

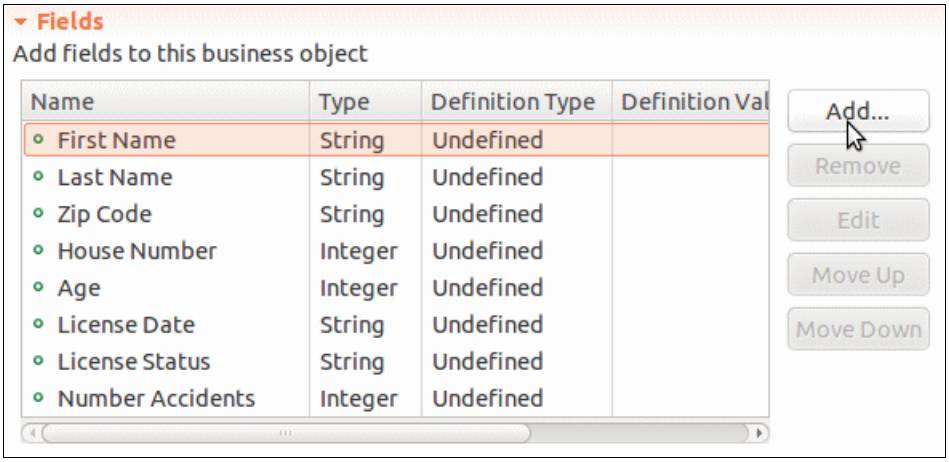


Figure 4-7 All required fields in the Driver business object

4.2.4 Creating the event

Next, you create the business event that is received from CICS. The event contains an event object that stores the information that is required to populate the business objects.

> **Request_data event object:** When emitting events from CICS, you can have only one event object in which all the data fields are stored. This event object is called *Request_Data*. This limitation means that the fields for both the driver and the vehicle are placed in the Request_Data event object and then mapped to the correct fields of the necessary business object.

Creating the event

To create the event, follow these steps:

1. Right-click the **RequestForQuote** project, and select **New** → **Event**.
2. Select **Create a blank event**, and click **Next**.
3. Enter the name `Request` for the event, and click **Next**.
4. Leave the connector set to **None**, and click **Finish**.

> **Configuration note:** You configure this connector for the event in 4.2.7, “Configuring the technology connectors” on page 87.

Creating the event object

To create the event object, follow these steps:

1. Right-click the **Request Event** and select **New** → **Event Object**. Then, select **Add a new blank Event Object**.
2. Enter the name `Request_Data`, and select the **RequestForQuote** event project and **Request** event, as shown in Figure 4-8.

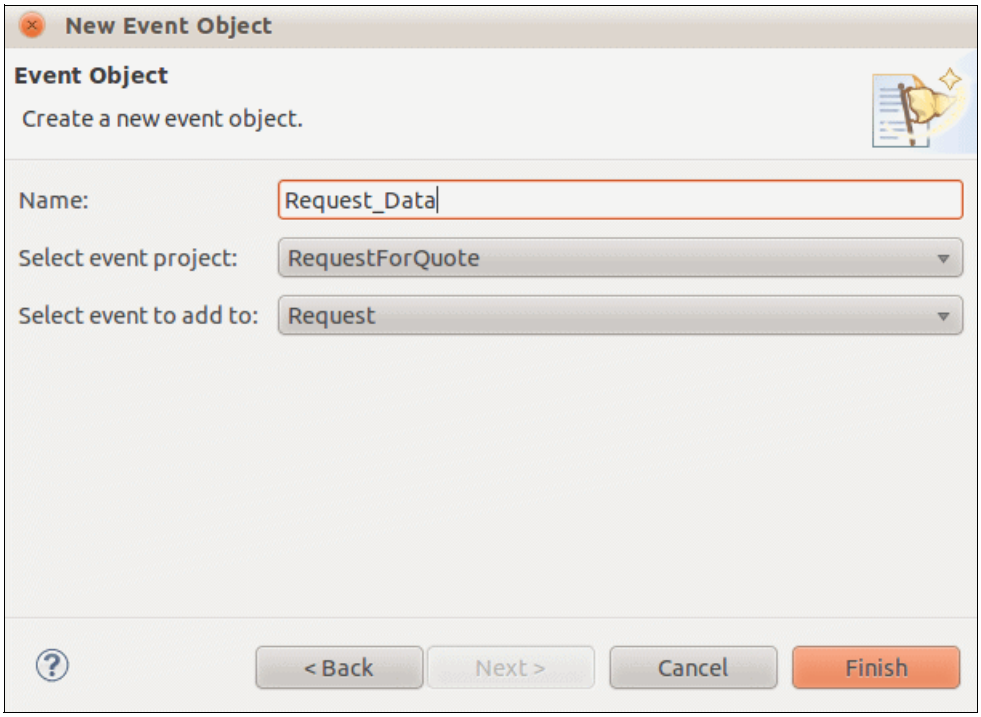


Figure 4-8 New Event Object

3. Click **Finish**, and the event object is generated.

Next, you must add the required fields to the Request_Data event object.

Creating the event object fields

The data fields for both the driver and the vehicle are added to the Request_Data event object, because events emitted from CICS can contain only one event object. This scenario requires the event to contain both the driver and vehicle information. This scenario uses the

mapping capabilities in the Event Designer to map the fields from the Request_Data event object into the correct Vehicle and Driver business objects.

To create the event object fields, follow these steps:

1. The Request_Data event object opens in the Event Object Editor after it is created in the previous steps. If it does not open, double-click the **Request_Data** event object to open it.
2. Click **Add** in the Fields section to add a field to the event object.
3. Specify the name of the first field as `First_Name`, and then select the **String** data type.
4. Click **Finish** to create the first field.
5. Repeat this process for all the fields that are described in Table 4-1 to complete a full list of fields, as shown in Figure 4-9.

Table 4-1 Fields required in the Request_Data event object

Field Name	Field Data Type
First_Name	String
Last_Name	String
ZIP_Code	String
House_Number	Integer
Age	Integer
License_Date	String
License_Status	String
Number_Accidents	Integer
ID	String
Make	String
Model	String
Value	Integer
Type	String

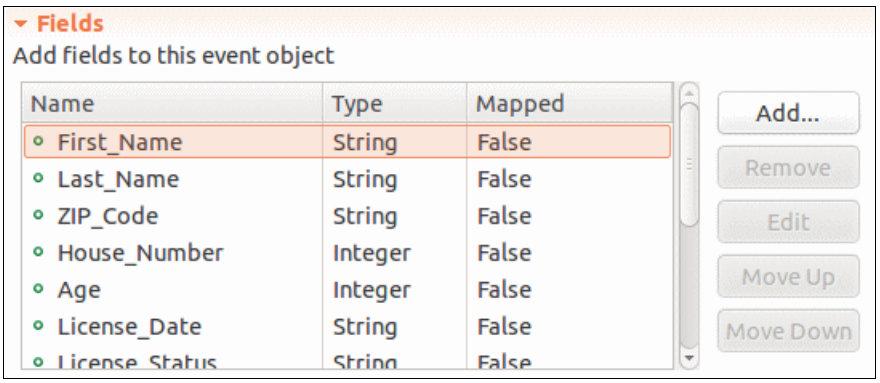


Figure 4-9 The event object with the completed list of fields

Mapping the event object fields to the business objects

Now that the data fields are created in the Request_Data event object, they must be mapped into the Driver and Vehicle business objects so that they can be used during processing when the event is received:

1. To map the event object fields to the appropriate business object, click **Add** in the Field Constructors view of the Overview tab of the Request_Data event object, as shown in Figure 4-10.

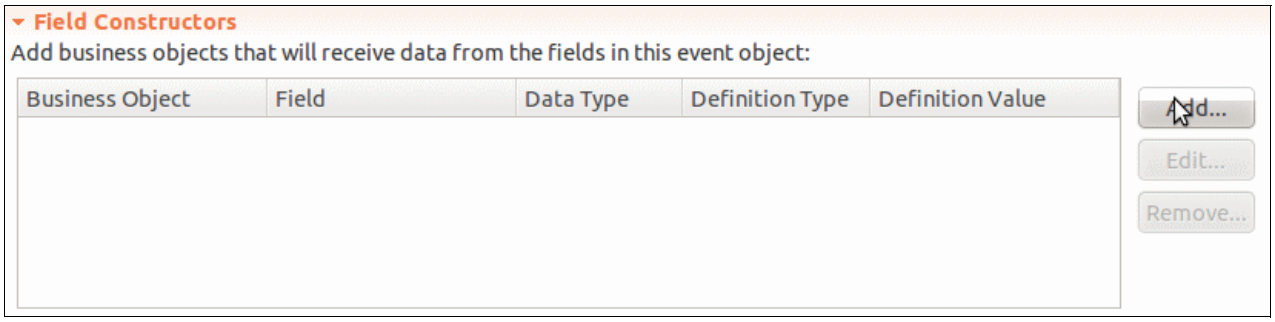


Figure 4-10 Add new field constructors to this event object

2. Select both **Driver** and **Vehicle** from the new Select the business objects to construct window, as shown in Figure 4-11, and click **Finish**.

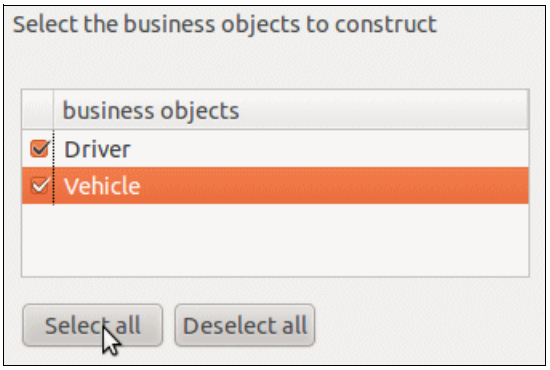


Figure 4-11 Selecting the business objects

3. Next, select the **Field Constructor** tab at the bottom of the Event Object Editor window.
4. For each Business Object field, select the Definition Type as **Field**, and select the appropriate event object field to which to map, as shown in Figure 4-12 on page 84, for the First Name field of Driver.

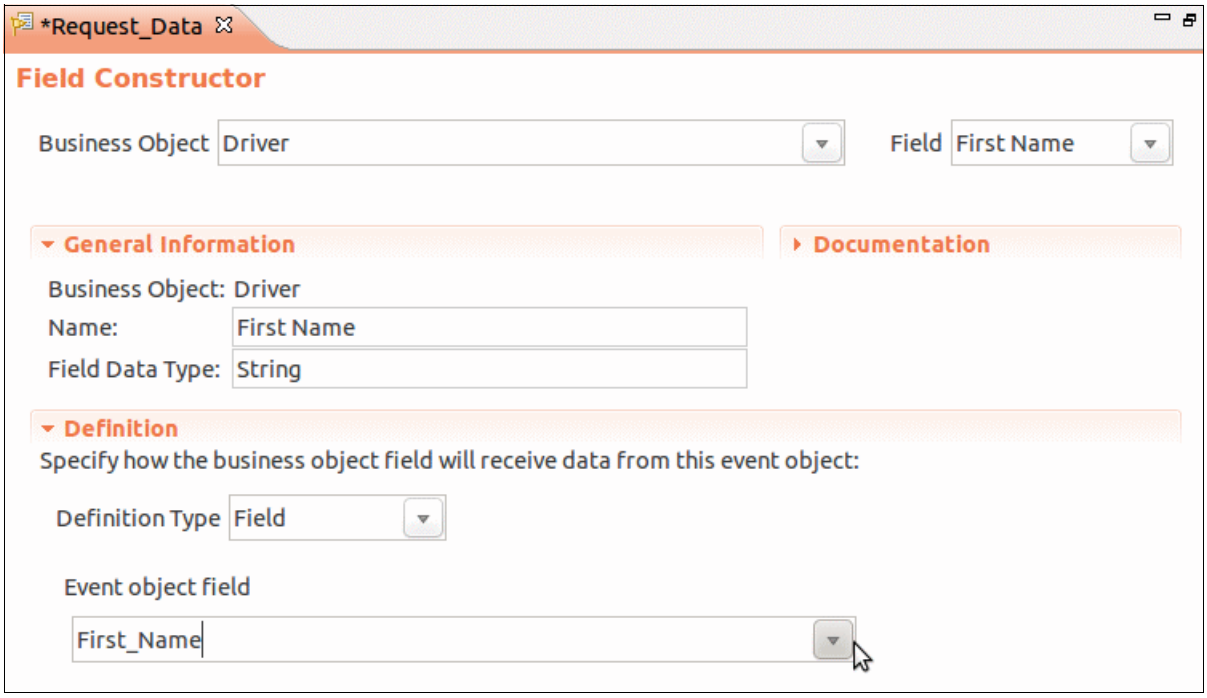


Figure 4-12 Mapping the First Name field of the Driver business object to the First_Name field of the event object

5. Repeat this process for the fields of both the Driver and Vehicle business objects, as shown in Table 4-2.

Table 4-2 Business object fields to which the event fields must map

Business object	Business object field	Event field name to map to
Driver	First Name	First_Name
Driver	Last Name	Last_Name
Driver	ZIP Code	ZIP_Code
Driver	House Number	House_Number
Driver	Age	Age
Driver	License Date	License_Date
Driver	License Status	License_Status
Driver	Number Accidents	Number_Accidents
Vehicle	ID	ID
Vehicle	Make	Make
Vehicle	Model	Model
Vehicle	Value	Value
Vehicle	Type	Type

4.2.5 Creating the action

The action that fires when the event pattern is identified is now created from the business object.

Creating the action

To create the action, follow these steps:

1. Right-click the **RequestForQuote** project, and select **New** → **Action**.
2. Select **Create a blank Action**, and click **Next**.
3. Enter the name `FollowUp` for the Action, and click **Next**.
4. Leave the connector set to `None`, and click **Finish**.

Creating the action objects

To create the action objects, follow these steps:

1. Right-click the **Driver** business object, and select **Create Action Object** from this business object.
2. Enter the name `Driver`, and select the **RequestForQuote** project and **FollowUp** action, as shown in Figure 4-13.

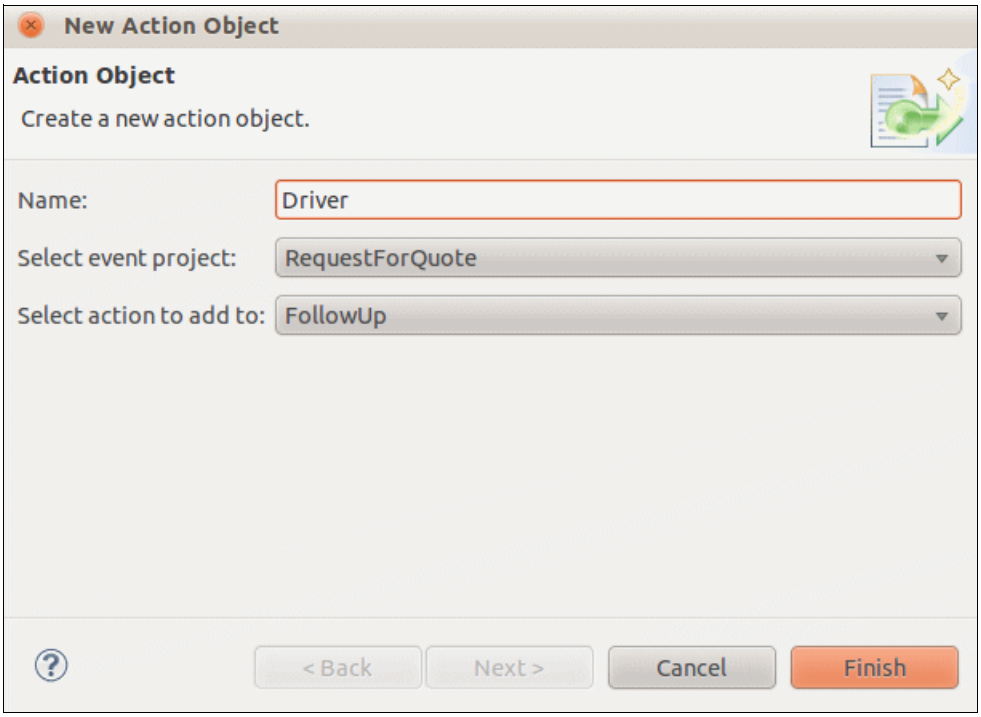


Figure 4-13 Creating the Action Object from the business object

3. Click **Finish**, and the action object is generated. Using this method ensures that the action object has all of the required field constructors to map the data from the business object to the action. This technique also avoids the manual mapping steps required for the event object fields.
4. Repeat this step to create the Vehicle Action Object, specifying Vehicle as the name for the new action object.

4.2.6 Creating the event rule

All the necessary artifacts for the event application are created. Event logic can now be written. It is triggered by events. Then, it acts on the business objects and emits actions.

Creating the event rule

The event logic is stored in an event rule. To create the event rule, follow these steps:

1. To create the event rule, right-click the **RequestForQuote** project, and select **New** → **Event Rule**.
2. Enter `IdentifyFollowUp` as the name of the event rule, and click **Next**.
3. Select the **Request** event as the event to trigger this event rule, and click **Next**.
4. Select the **ID of the vehicle** as the context relationship, and click **Finish**.

Using a context relationship for the event rule

In this scenario, Decision Server Events must identify whether the same customer requested more than three quotes in the same hour. To enable this function, a context relationship must be defined so that Decision Server Events has a field that it can use to identify events from the same customer.

For this scenario, the ID of the vehicle was selected as the unique identifier to allow us to identify the pattern of more than three quotes within the same hour.

Writing the event rule

The event rule is entered directly in to the Event Rule editor, as shown in Figure 4-14.

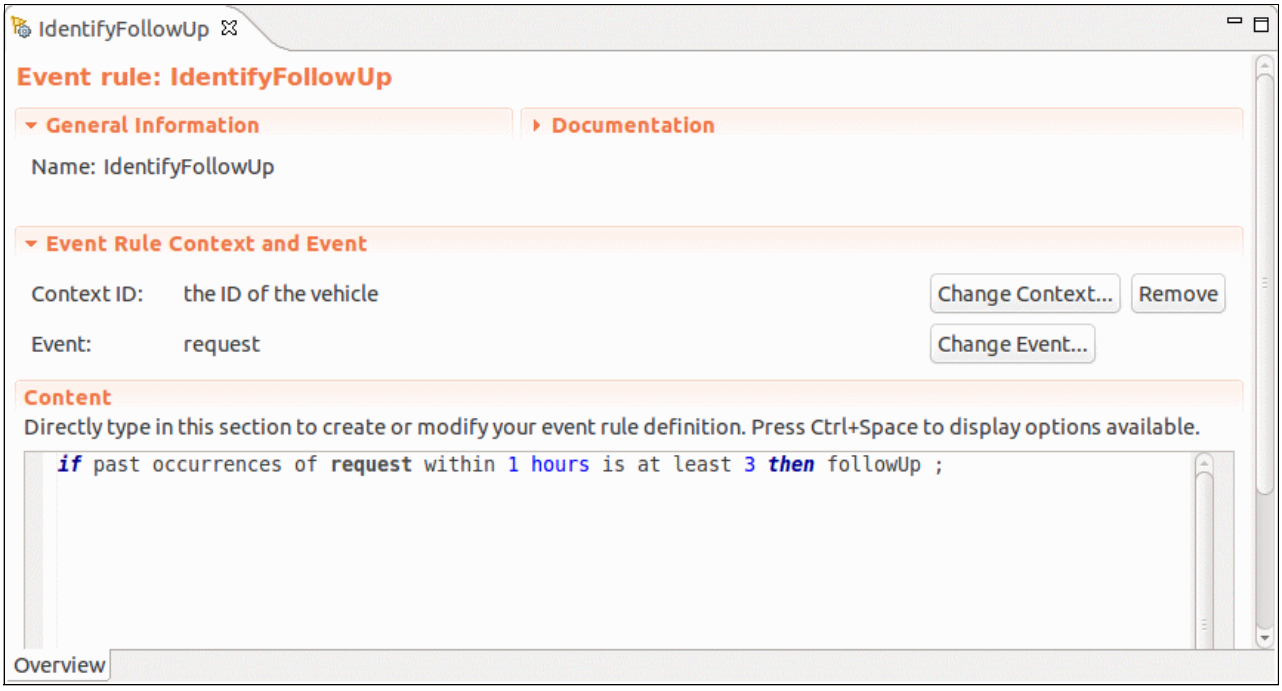


Figure 4-14 Entering the event logic in the Event Rule editor

The event uses the following logic:

```
if past occurrences of request within 1 hour is at least 3 then followUp;
```

4.2.7 Configuring the technology connectors

The final authoring stage for the event application is to configure the event and action technology connectors (connectors). The connectors define which protocol the Decision Server Events run time uses to receive the event and send the action.

Decision Server Events supports the following multiple event and action connectors:

- Available event connectors:
  - Email: Receive an event by POP3
  - File: Receive an event when a file is added or modified in a folder
  - FTP: Receive an event when a file is added or modified on an FTP site
  - HTTP: Receive an event by HTTP
  - Java Database Connectivity (JDBC): Receive an event by executing an SQL statement
  - Java Message Service (JMS): Receive an event directly from a JMS destination
  - SOAP: Receive events by SOAP
- Available action connectors:
  - Email: Send an action by Simple Mail Transfer Protocol (SMTP)
  - File system: Save the action as a file in a folder
  - FTP: Save the action as a file on an FTP site
  - HTTP: Send the action by HTTP
  - JDBC: Send an action by executing an SQL statement
  - JMS: Send an action directly to a JMS destination
  - Representational State Transfer (REST): Send an action to an application by using the REST interface
  - SOAP: Invoke a web service by using SOAP 1.1
  - User console: Send an action to the user console for review and response

Configuring the event connector

The Request Event is received from CICS using the HTTP connector:

1. Double-click the **Request** event to open the Event editor.
2. Select the connector tab to navigate to the connector panel, as shown in Figure 4-15.

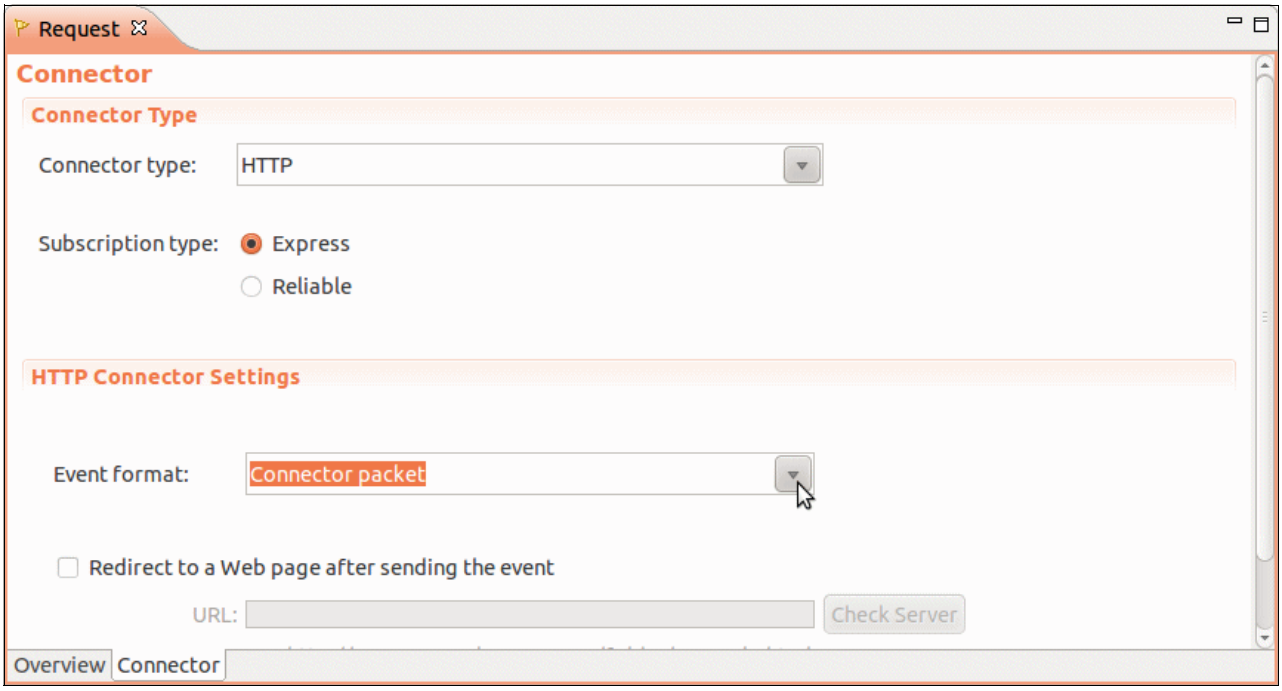


Figure 4-15 The Connector panel of the Event editor

3. From the Connector Type drop-down menu, select **HTTP** as the connector type for the Request event.
4. From the Event format drop-down menu, select the **Connector packet** option.
5. Save the changes, and close the Event editor.

Configuring the action connector

For this scenario, you do not configure a connector for the FollowUp action. Instead, the Events tooling is used to verify that the action fired when the business rule is met.

In an actual scenario, an appropriate connector is used to send the generated action to the required system. For example, the FollowUp action can be delivered through email to the sales team to notify them to call the customer to follow up on the customer's requests for a quotation.

4.3 Deploying the event application to the event run time

The Decision Server events run time on z/OS is a WebSphere application that is hosted inside the WebSphere Application Server for z/OS.

4.3.1 Creating the event runtime connection

To deploy the event application to Decision Server Events, the Event Designer creates an event runtime connection. Using this connection, you can both deploy and view assets on the server.

To create the event runtime connection, follow these steps:

1. Select the **Event Runtimes** tab, and right-click in the empty white space.
2. Click **New runtime connection**, and enter the server information, as shown in Figure 4-16:
   - For Host name, type `local host`.
   - For Port, type `9080`.

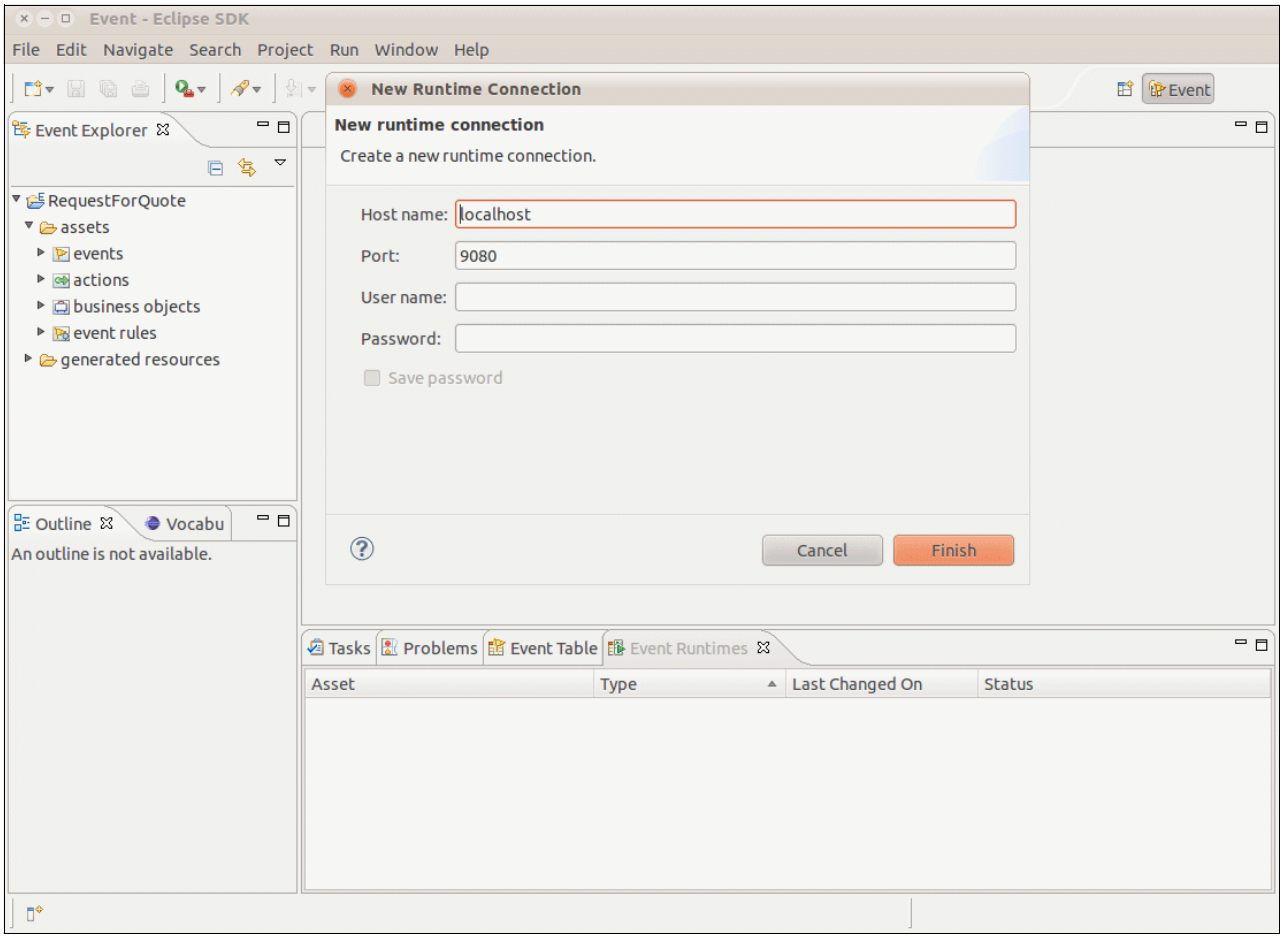


Figure 4-16 Entering the information for a new runtime connection

3. Click **Finish** to create the connection and connect the Event Designer to the Decision Server Events run time.
4. Figure 4-17 shows the Event Designer connected to the Decision Server Events run time, which has no assets deployed.

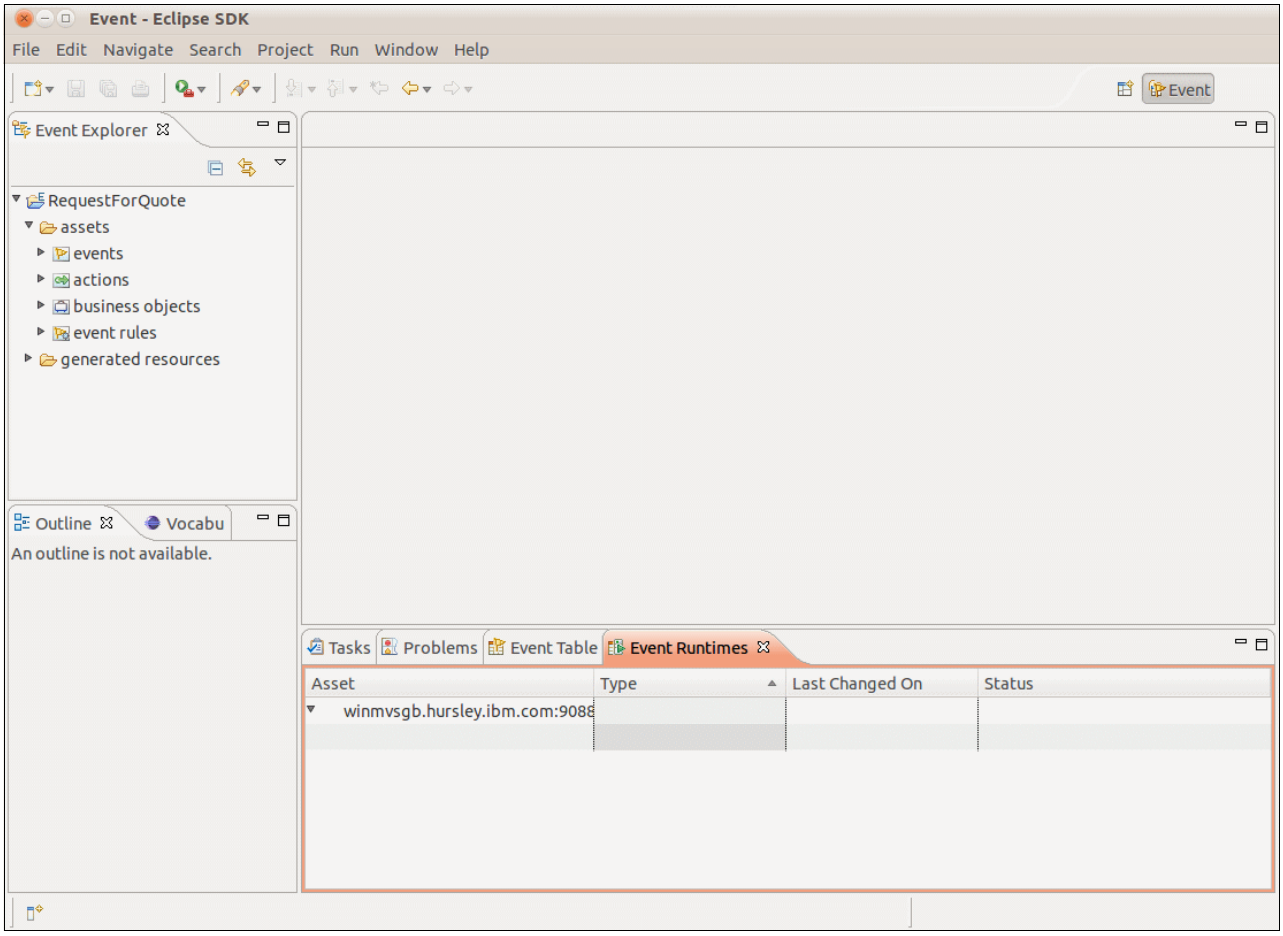


Figure 4-17 The new Event Runtimes tab with an empty repository

4.3.2 Deploying the event project to the event run time

To deploy the event project to the event run time, follow these steps:

1. Right-click the **RequestForQuote** event project, and select **Deploy**.
2. Select **Deploy all assets**, and click **Next**.
3. Select **Use a known runtime**, and highlight the recently created runtime definition.
4. Click **Finish** to deploy the assets.

After the deployment completes, the assets are shown in the Event Runtimes tab, as shown in Figure 4-18.

Tasks | Problems | Event Table | Event Runtimes

Asset	Type	Last Changed On	Status
winmvsgb.hursley.ibm.com:9088			
FollowUp	Action	2011-12-06T10:31:01Z	In project 'RequestForQuote'
Driver	Action object	2011-12-01T13:32:28Z	In project 'RequestForQuote'
Vehicle	Action object	2011-12-01T13:32:37Z	In project 'RequestForQuote'
Driver	Business object	2011-12-01T12:22:59Z	In project 'RequestForQuote'
Vehicle	Business object	2011-12-01T12:20:50Z	In project 'RequestForQuote'
Request	Event	2011-12-06T10:24:50Z	In project 'RequestForQuote'

Figure 4-18 The deployed assets as shown in the Event Runtimes tab

Because this event project uses the HTTP connector, you now see that a new application is automatically deployed to your WebSphere Application Server, as shown in Figure 4-19.

wbehttpconnector	
wberuntimeear	

Figure 4-19 The wbehttpconnector web application on WebSphere Application Server

The HTTP event connector is used to receive an event by using an HTTP POST and to send an action by using HTTP.

Configuring the HTTP connector for security

When Decision Server Events is installed into a WebSphere Application Server with security enabled, you must specify a user role mapping for the wbehttpconnector application. The user role mapping determines which users have permission to send events through HTTP to the connector.

To specify the user role mapping using the WebSphere Application Server administrative console, complete the following steps:

1. Log in to the WebSphere Application Server administrative console.
2. In the menu, expand **Applications** → **Application Types**, and click **WebSphere enterprise applications**.
3. In the main panel, click the HTTP connector application, **wbehttpconnector**.
4. Under Detailed Properties, click **Security role to user/group mapping**. Select the **HTTPEventConnectorUser** role, and map one or more users to the role. To allow unrestricted access when security is enabled, map the role to the **Everyone** special subject.
5. Click **Save**.

These steps are required only when an event project that uses the HTTP connector is first deployed. When subsequent projects are deployed, the existing mappings are maintained.

4.4 Emitting events from CICS

The event project is now deployed to the event run time, and Decision Server Events is waiting to receive events through HTTP. For this example, we configure CICS to send the Request event each time that a customer requests a quotation.

4.4.1 CICS event support

Given the considerable amount of business processing that is performed in CICS systems across the world (over 30 billion transactions a day), CICS is a significant source of business events. Events can provide enhanced business flexibility and help to meet governance and compliance regulations.

Using event specifications that are defined to CICS, events can be captured from existing business application programs without altering the original code. These capture points include relevant CICS API calls and when a program starts. Each time that a program executes a capture point, CICS checks each enabled capture specification that matches the capture point.

Each matching capture specification contains optional filters to compare against the application context, several command options on the API call, and data passed on the API call. If the filters match, CICS collects the payload information from information sources described in the capture specification, enriches it with context data, and then queues the event for dispatch so that the application can continue to execute quickly.

4.4.2 CICS Event Binding Editor

The CICS Event Binding Editor in the IBM CICS Explorer® is used to create an event binding that contains an event specification for our program. The following instructions contain the information that is required to create the CICS Bundle project for this scenario. For information about CICS event processing, see *Implementing Event Processing with CICS*, SG24-7792:

http://www.redbooks.ibm.com/abstracts/sg247792.html?Open

You can download the CICS Explorer onto your workstation through the CICS Explorer website:

http://www-01.ibm.com/software/htp/cics/explorer/

4.4.3 Creating the CICS Bundle project

To create the CICS Bundle project, follow these steps:

1. Start the CICS Explorer from the location to which it was downloaded.
2. Switch to the Resource perspective in CICS Explorer.
3. Create a CICS Bundle project by selecting **Explorer** → **New Wizards** → **CICS Bundle project**.
4. Enter `RequestForQuote` as the name, and click **Finish**.

4.4.4 Creating the event binding

An *event binding* defines a business event to CICS. An event binding can be created using the CICS Event Binding Editor by business analysts and developers or by an application analyst in response to a business requirement. An event specification describes an event and its processing in natural language.

To create the Event Binding, follow these steps:

1. Right-click the **RequestForQuote** project, and select **New** → **CICS Event Binding**.
2. Enter `quotation` as the name, and click **Finish**.

A new event binding is created and an error message is also displayed, as shown in Figure 4-20. A new event specification must now be created.

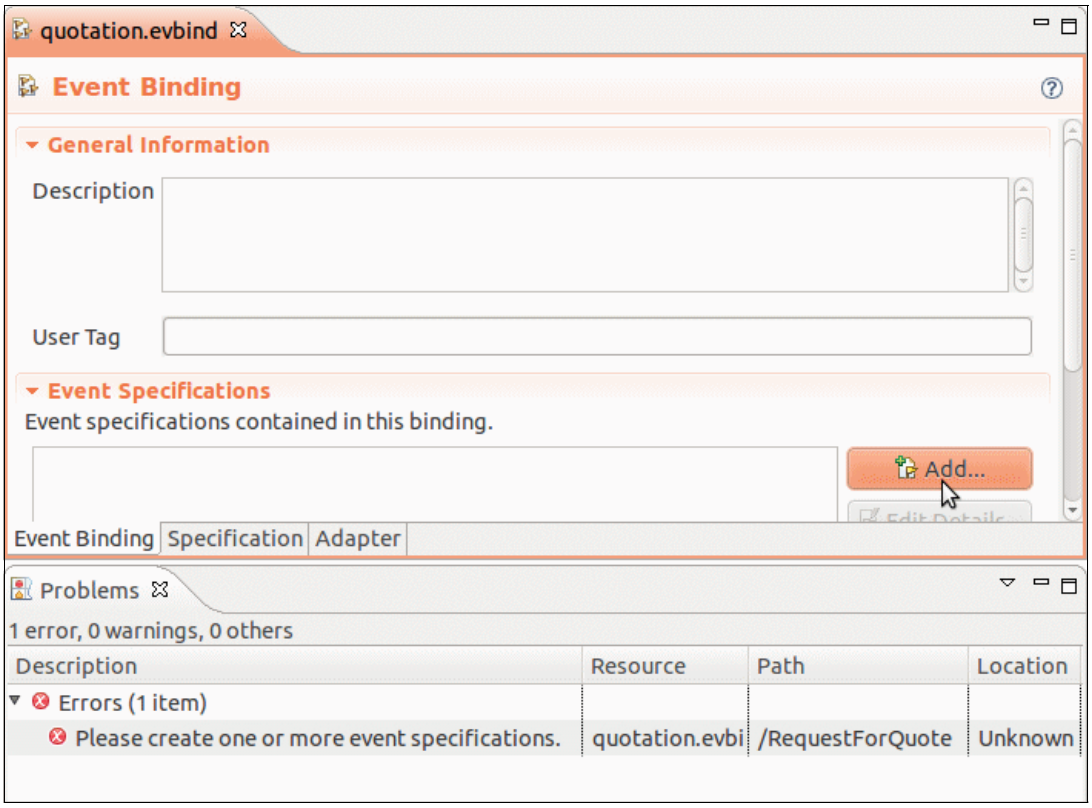


Figure 4-20 The new event binding with an error message displayed

4.4.5 Creating the event specification

The *event specification* describes to CICS the information that is contained in the event that is emitted to the Decision Server Events run time.

To create the event specification, follow these steps:

1. Click **Add**, as shown in Figure 4-20.
2. Enter `Request` as the name for the specification, and click **Finish**.

3. The event specification is now created and must be edited. Click **Edit Details**, as shown in Figure 4-21.

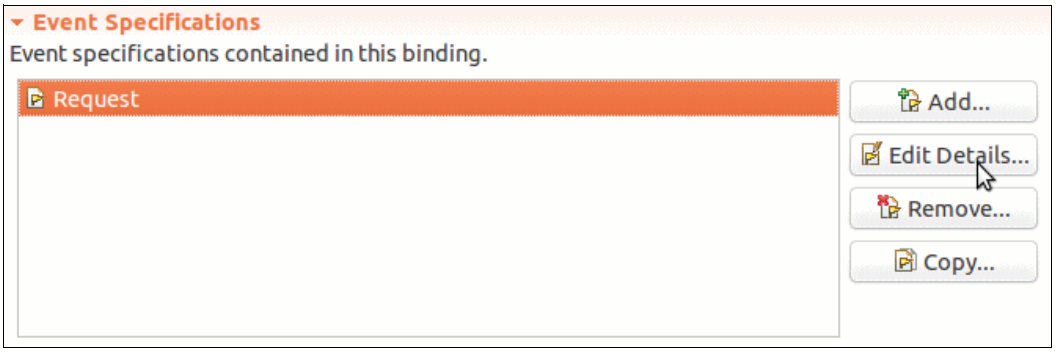


Figure 4-21 The newly created event specification

4. You must define the fields that CICS emits in this event. To add the first field, click **Add** in the Emitted Business Information table, as shown in Figure 4-22.

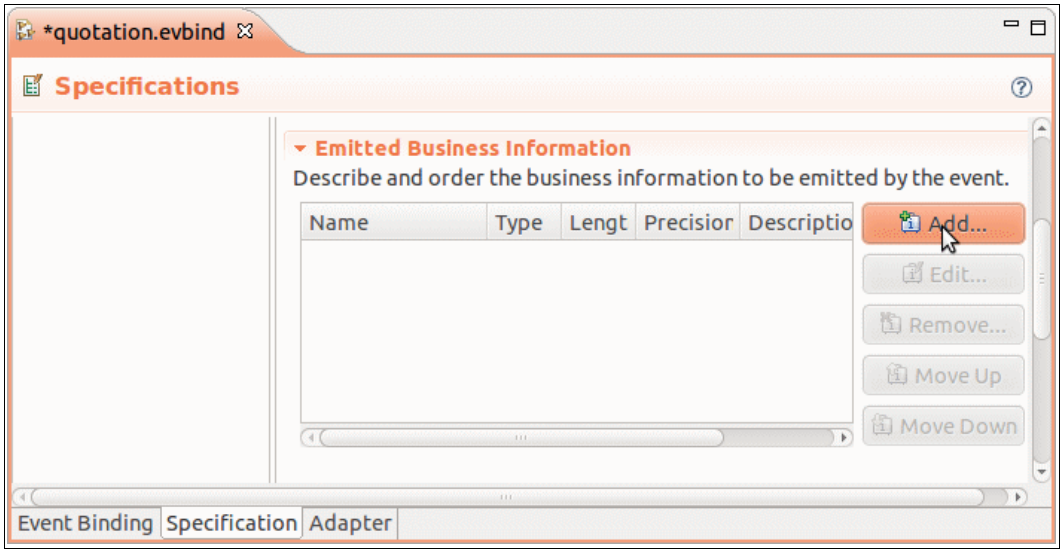


Figure 4-22 The emitted business information is created

5. Enter the first field as `First_Name` with the data type as `Text` and click **OK**.
6. Repeat step 5 with the field names and data types that are listed in Table 4-3.

Table 4-3 The fields defined in the event specification

Field name	Field data type
First_Name	Text
Last_Name	Text
ZIP_Code	Text
House_Number	Numeric
Age	Numeric
License_Date	Text

Field name	Field data type
License_Status	Text
Number_Accidents	Numeric
ID	Text
Make	Text
Model	Text
Value	Numeric
Type	Text

4.4.6 Creating the capture specification

Now that the event specification is complete, we can tell CICS how to capture the event and how to emit it to the Decision Server Events run time.

To create the capture specification, follow these steps:

1. Click **Add a Capture Specification** on the Event specification tab.
2. Enter `Request` the name as and click **Finish**.
3. Click the newly created **Request** capture specification.
4. Scroll down and select **WRITEQ TS** as the Application Capture Point.
5. Next, click the **Filtering** tab at the top. In the Event Options section, specify that the QNAME **Equals** `REQUESTS`, as shown in Figure 4-23.

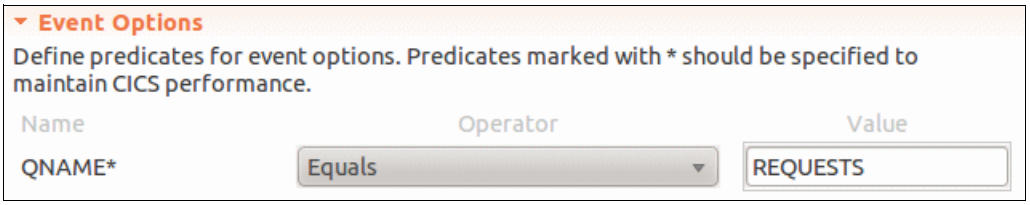
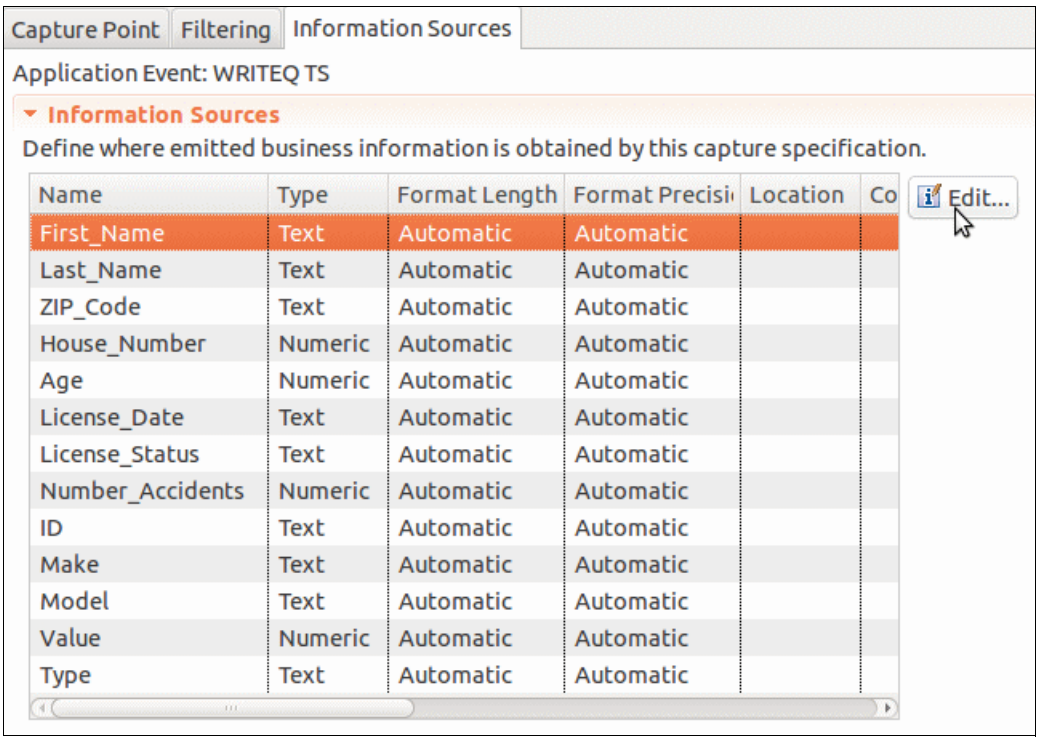


Figure 4-23 Filtering which TS Queue causes the event to be emitted

6. Click the **Information Sources** tab to display the event data fields. The COBOL copybook is used to describe how the data that is written to the TS queue is mapped in to the event fields.
7. Select the first row in the table, the **First_Name** field, and click **Edit**, as shown in Figure 4-24.



Name	Type	Format Length	Format Precisi	Location	Co
First_Name	Text	Automatic	Automatic		
Last_Name	Text	Automatic	Automatic		
ZIP_Code	Text	Automatic	Automatic		
House_Number	Numeric	Automatic	Automatic		
Age	Numeric	Automatic	Automatic		
License_Date	Text	Automatic	Automatic		
License_Status	Text	Automatic	Automatic		
Number_Accidents	Numeric	Automatic	Automatic		
ID	Text	Automatic	Automatic		
Make	Text	Automatic	Automatic		
Model	Text	Automatic	Automatic		
Value	Numeric	Automatic	Automatic		
Type	Text	Automatic	Automatic		

Figure 4-24 Editing from where the event fields receive their data

8. In the new window, select **FROM** in the Available Data view. Click **Select from imported language structure**, as shown in Figure 4-25.

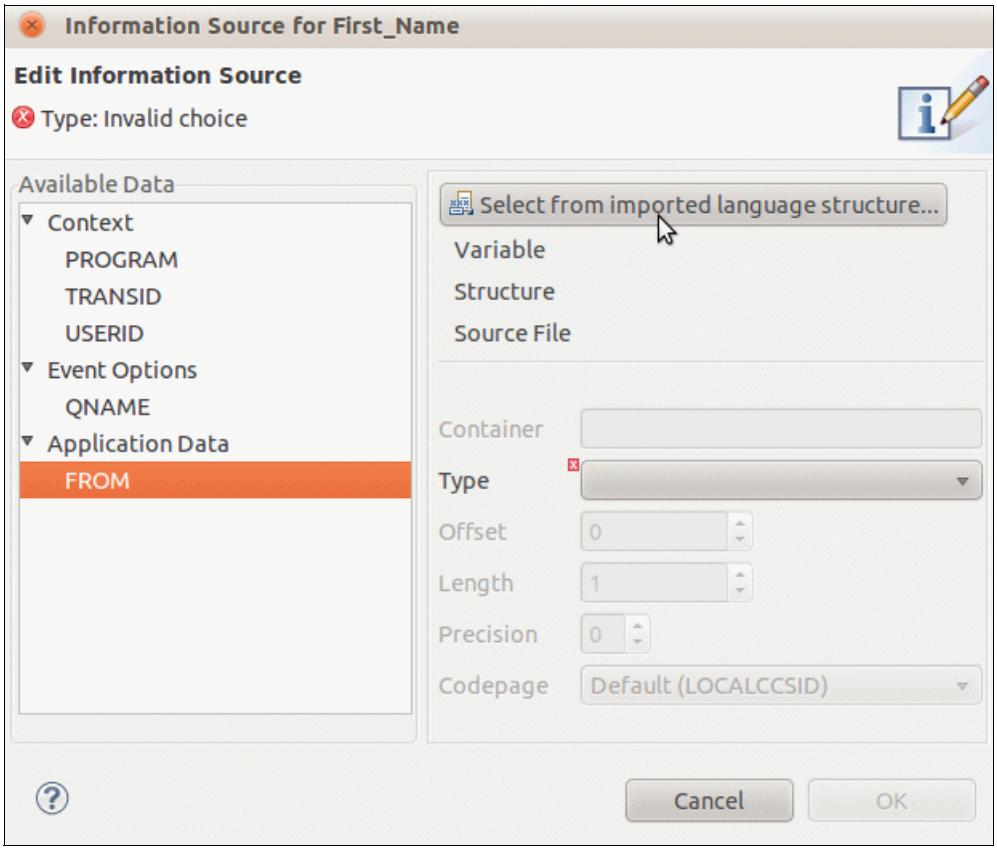


Figure 4-25 Importing the COBOL copybook

9. Click **Choose Language Structure File**, navigate to the COBOL copybook that is used for this scenario, and click **OK**.
10. Ensure that the Source Language is specified as COBOL and click **OK**.

11. In the new Language Structure window, expand the **driver** section and select the **first_name** field, as shown in Figure 4-26.

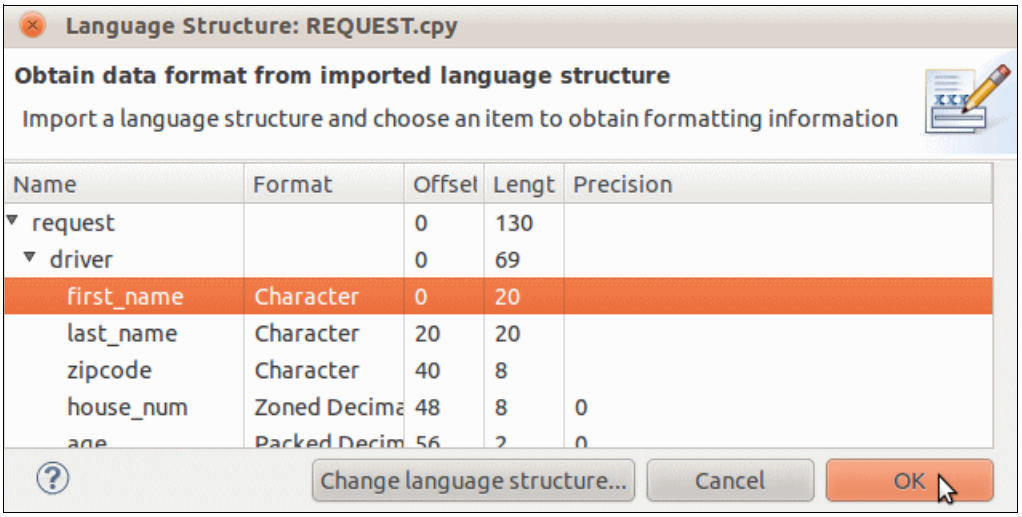


Figure 4-26 The COBOL copybook is used to define the event data mapping

12. Click **OK** to finish the mapping of the COBOL data to the event field.
13. Repeat these steps for the remaining fields on the Information Sources tab. Ensure that the correct field is selected from the COBOL copybook for the event field that you are editing.

4.4.7 Defining the adapter

The adapter defines the protocol that CICS uses to send the event to the Decision Server Events run time. In this scenario, the HTTP adapter is used to send the event. To configure the adapter, follow these steps:

1. Select the **Adapter** tab of the Event binding editor. Figure 4-27 on page 99 shows the completed tab.
2. Click **Use an adapter defined here**.
3. Select **HTTP** from the list of adapters.
4. Enter the name `DSEVENTS` as the Urimap. This name is defined to CICS in a later step.
5. For the Data Format, select **WebSphere Business Events (XML)** from the drop-down list.

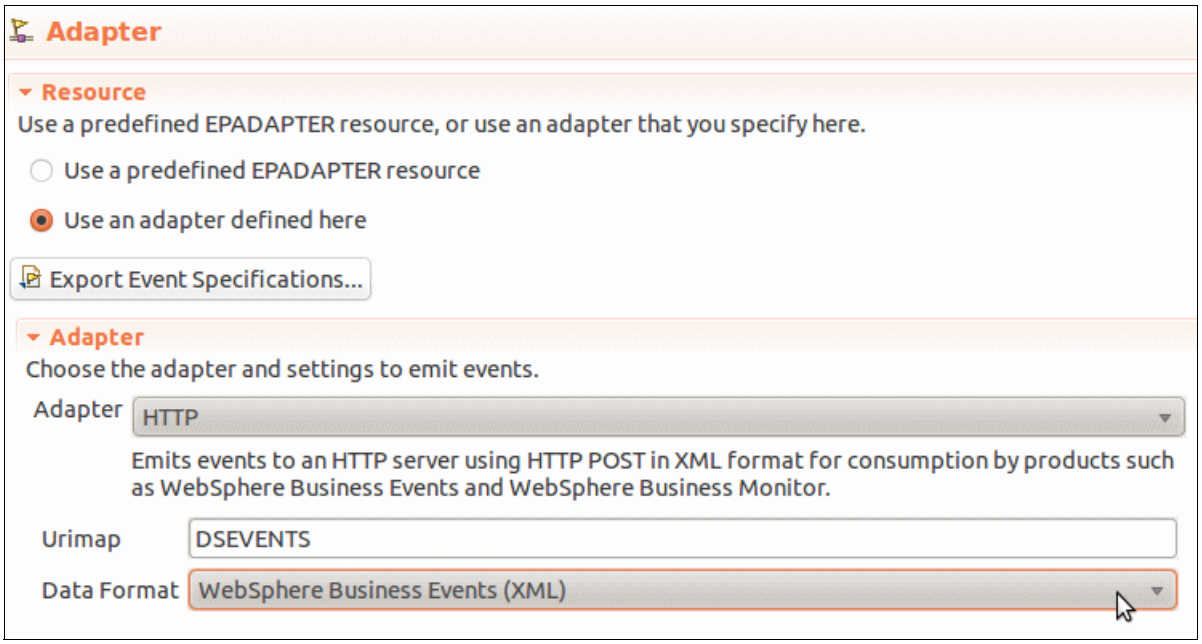


Figure 4-27 The completed Adapter specification that CICS uses to transmit the events

> **Note:** The adapter is defined in the event bundle here for a quick deployment. In an actual scenario, it is recommended that the adapter is defined as an EPADAPTER in CICS. This approach separates the creation and management of the adapter from the event binding file.

6. Save the event binding and close the Event binding editor.

4.4.8 Deploying the bundle to CICS

The RequestForQuote CICS bundle now contains the completed Request event binding. The event binding describes the fields that are emitted in the Request event, how the event is triggered, and how data from the COBOL application is mapped into the event fields.

The next step is to configure CICS to emit the Request event to the Decision Server Events run time each time that a customer requests a quotation.

Enabling CICS for events

First, CICS must be enabled for event processing. Event processing is enabled by default when a START=INITIAL or START=COLD parameter is used during the startup of your CICS TS V4.1 or V4.2 system. When a START=WARM or START=EMERGENCY parameter is used, the settings from the previous run of CICS are used. So, unless the setting changed, event processing is enabled.

You can inquire on EVENTPROCESS to check whether event processing is enabled, for example, by using the CICS Explorer. You can stop and start event processing from the IBM CICS Explorer, the CICSplex SM Web User Interface, or the CICS SPI or API command. You might want to stop event processing for an upgrade or system maintenance, and then start event processing again.

Deploying the bundle

The CICS bundle must be transferred to a directory in the z/OS Distributed File Service (zFS) for CICS to use it. The CICS Explorer provides the required functionality to transfer the CICS bundle to the zFS.

To transfer the bundle, follow these steps:

1. Ensure that you configure an FTP connection under **Window** → **Preferences** → **CICS Explorer** → **Connections**.
2. Next, right-click the **RequestForQuote** bundle, and select **Export Bundle Project to z/OS UNIX File System**.
3. Navigate to a directory to which you have write access, typically in your home directory, for example, `/u/hiscm/RequestForQuote`, and click **Finish**.

The CICS Explorer deploys the bundle to the specified directory.

Creating a bundle definition

After the bundle is deployed to the zFS, its location must be defined to CICS by using a bundle definition (BUNDEF).

To create the bundle definition, follow these steps:

1. Select **Explorer** → **New Wizards** → **Other**.
2. Expand the **CICS Definitions** folder, select **Bundle Definition**, and click **Next**. The completed panel is shown in Figure 4-28 on page 101.
3. Enter the group as `WODMEV`.
4. Enter the name of the bundle definition as `RFQ`.
5. Enter the Bundle Directory as the same location that you specified previously, for example, `/u/hiscm/RequestForQuote`.
6. Click **Finish** to complete the creation of the bundle definition.

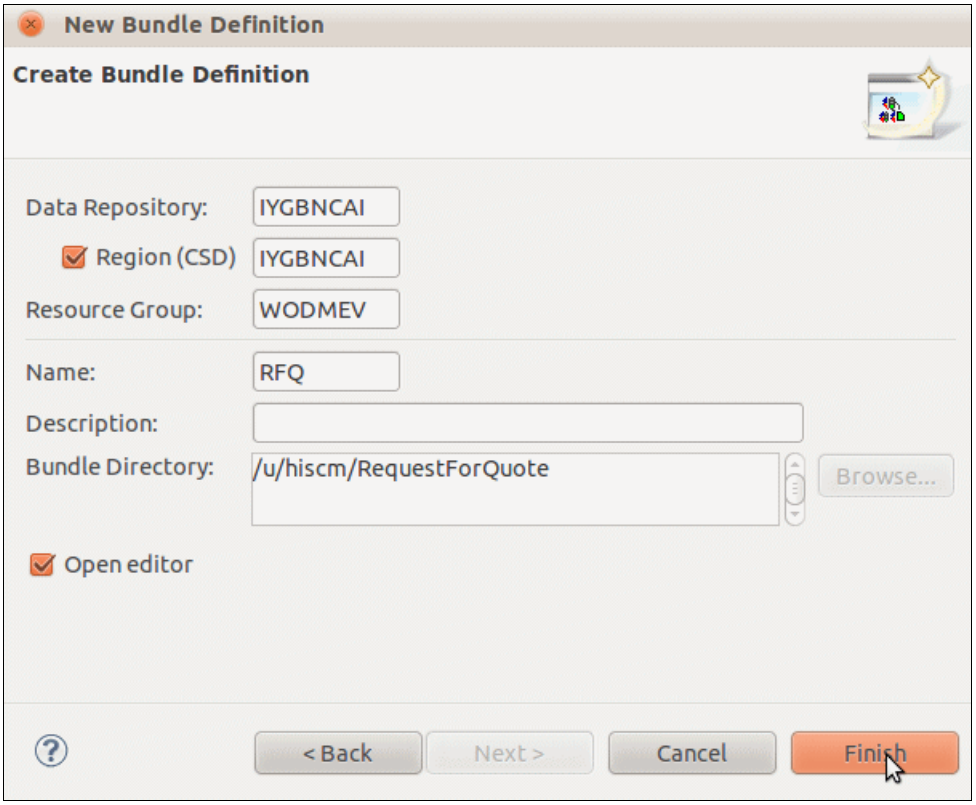


Figure 4-28 The completed bundle definition

Now that the bundle definition is created, it must be installed into CICS. Follow these steps to install the new bundle definitions:

1. Open the Bundle Definitions view by selecting **Window** → **Show View** and select **Bundle Definitions**.
2. Right-click the newly created **RFQ** bundle.
3. Select **install**.

4. Then, select **IYGBNCAI**, which is the CICS into which to install the bundle, and click **OK**, as shown in Figure 4-29.

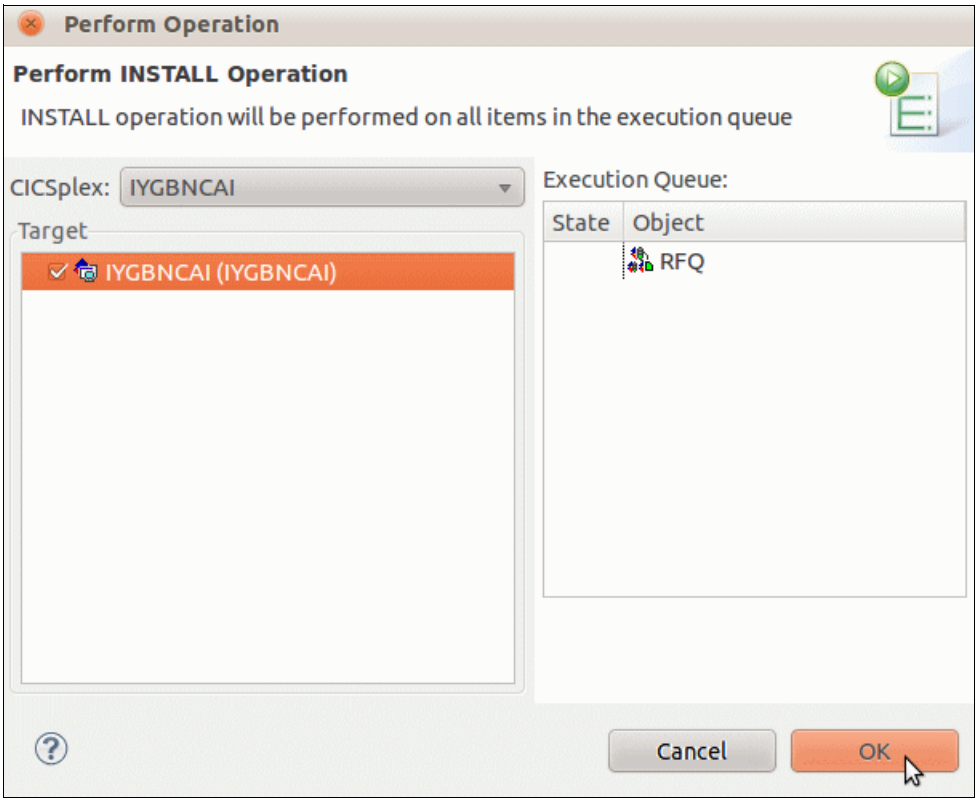


Figure 4-29 Installing the RFQ resource bundle

User ID access: Ensure that the user ID under which CICS is running has the appropriate access to the UNIX System Services directory to which the event binding is transferred.

Creating the URI mapping definition

The URI mapping definition DSEVENTS was used on the event adapter to tell CICS how to send the event to the Decision Server Events run time. Create this URI mapping definition:

1. Open the URI Mapping Definition view by selecting **Window** → **Show View** and select **URI Mapping Definition**.
2. Right-click in the new view and select **New**. The completed panel is shown in Figure 4-30 on page 103.
3. Enter the resource group as `WODMEV`.
4. Enter the name as `DSEVENTS`.
5. Enter the host where the Decision Server Events run time is located.
6. Enter the path as `/wbeca/HTTPEventConnector`.
7. Click **Client** and enter the port on which the Decision Server Events run time is listening.
8. Click **Finish**.

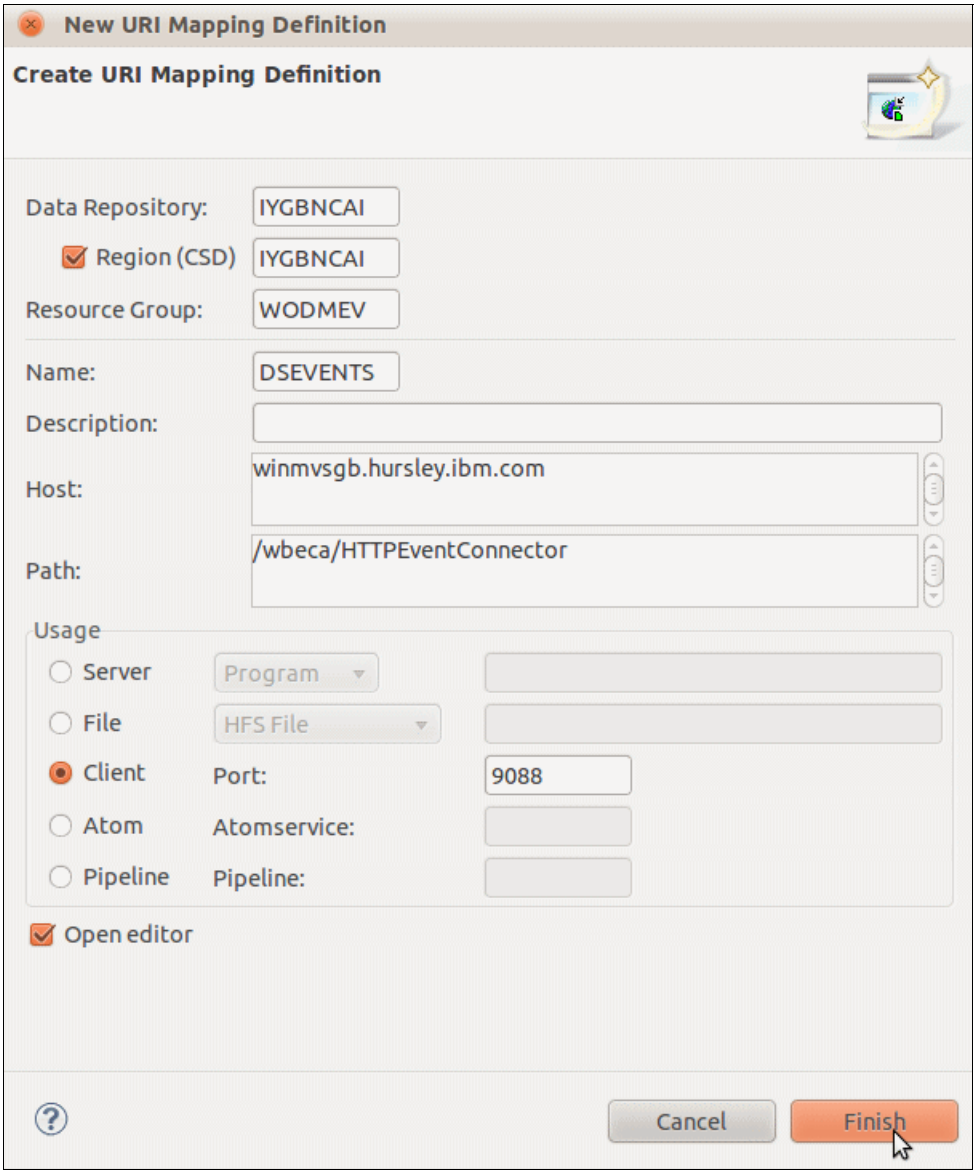


Figure 4-30 Creating the URI mapping definition

9. Right-click the newly created URI mapping definition and select **Install**.
10. Select the CICS to which to install the definition and click **OK**.

Creating the TS Queue

You can create the TS Queue using the CICS Explorer:

1. Open the **TS Model** view in the CICS Explorer.
2. Right-click in the white space in the view, and right-click **New**.
3. Fill in the details, as shown in Figure 4-31 on page 104:
   - For Data Repository, type `IYGBNCAI`.
   - Select **Region (CSD)** and **IYGBNCAI**.
   - For Resource Group, type `WODMEV`.
   - For Name, type `REQUESTS`.
   - For Prefix, type `REQUESTS`.

– Select **Open Editor**.

Click **Finish** to create the TS model definition.

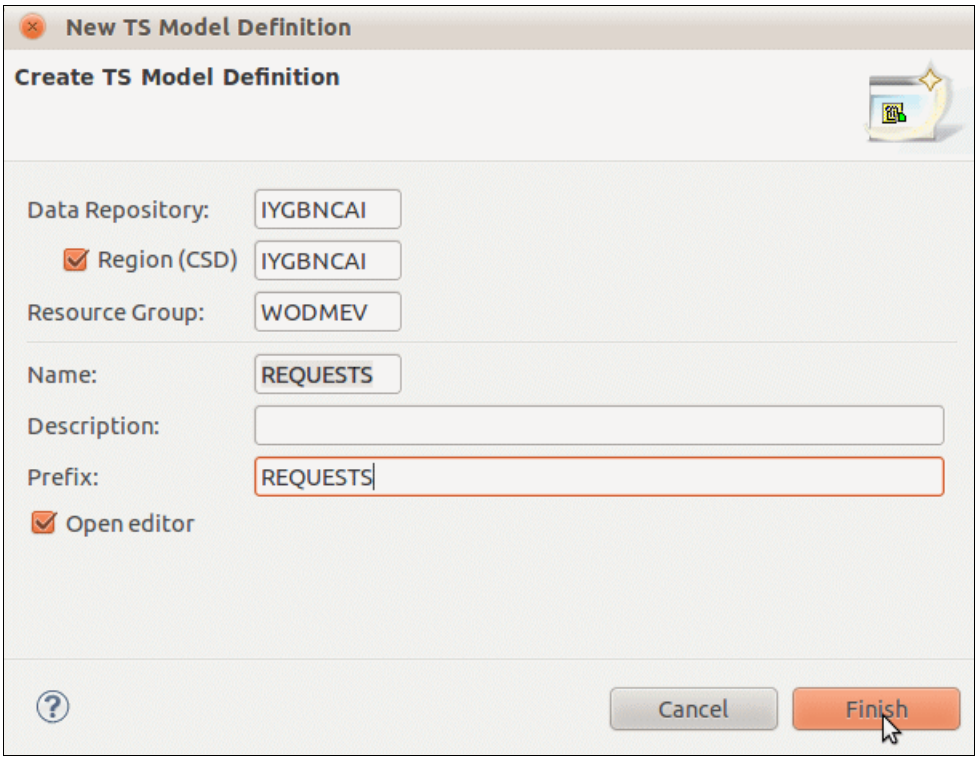


Figure 4-31 Creating the TS model definition

4. Finally, right-click the newly created definition and click **Install**.
5. Select your CICS and click **OK**.

4.5 Running the scenario

CICS is now configured to emit the Request event when any CICS program writes the Request COBOL structure to the REQUESTS TS Queue.

When this event is emitted, it is received by the Decision Server Events run time and processed using the Event rule.

4.5.1 Enabling history in the Decision Server Event run time

Before running the sample COBOL application, configure the Decision Server Event run time to record history. This function allows the run time to remember the received events and the actions that are sent.

Enabling this feature allows you to monitor the events sent from CICS. To enable history in the Decision Server Events run time, follow the instructions documented at this website:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.install.doc/doc/configuringwebspherebusinesseventsruntimetorecordhistory.html

4.5.2 Sample COBOL application to emit the Request event

To emit the Request event from CICS, the following COBOL application is provided. The application imports the Request copybook and fills out the necessary data fields. It then writes the Request data structure to the REQUESTS TS queue to trigger the event emission.

Example 4-2 The COBOL application that emits the Request event

```
CBL XOPTS(APOST SP COBOL3)
       PROCESS APOST
       PROCESS TRUNC(BIN) LIB RENT LIST
       PROCESS MAP XREF(FULL) NONAME

       IDENTIFICATION DIVISION.
       PROGRAM-ID. HBRRFQ.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       COPY REQUEST.
       PROCEDURE DIVISION.
           Move "Mark" TO FIRST-NAME
           Move "Hiscock" TO LAST-NAME
           Move "SO212JN" TO ZIPCODE
           Move 3 TO HOUSE-NUM
           MOVE 30 TO AGE
           MOVE "19201120" TO LIC-DATE
           MOVE "E" TO LIC-STATUS
           MOVE 0 TO NUMBER-ACCIDENTS
           Move "NCC-1701-D" TO VEC-ID
           MOVE "STARSHIP" TO MAKE
           MOVE "ENTERPRISE" TO MODEL
           MOVE 1000 To VEC-VALUE
           MOVE "SU" TO VEC-TYPE

           EXEC CICS WRITEQ TS QUEUE('REQUESTS') FROM (REQUEST)
           END-EXEC.
           GOBACK.
```

Compile this COBOL program using the Enterprise COBOL compiler for z/OS and install it into the same CICS into which the bundle definition was installed. Finally, define a transaction in CICS to run the HBRRFQ program.

This application is a stand-alone example of emitting an event from CICS using a COBOL application. You can combine this example with the rule application in Chapter 3, “Getting started with business rules” on page 27. That way, the Request event is only emitted when a successful rule execution is completed.

For example, Example 4-3 on page 106 shows a rule invocation which emits the Request event when a successful rule invocation is made.

Example 4-3 Combining a rule execution with an event emission

```
CALL 'HBRRULE' USING HBRA-CONN-AREA
           IF HBRA-CONN-COMPLETION-CODE NOT EQUAL HBR-CC-OK
               DISPLAY "invoke request rule failed"
           ELSE
               DISPLAY 'invoke request rule successful'
               EXEC CICS WRITEQ TS QUEUE('REQUESTS') FROM (REQUEST)
           END-IF
```

4.5.3 Emitting the event and firing the FollowUp action

In this section, we describe how to emit and receive an event, and check the FollowUp action.

Emitting the event

Ensure that the Decision Server Events run time starts and runs the transaction in CICS, which causes the HBRRFQ program to run. This program causes the Request event to be emitted with the data, as shown in the sample COBOL application in 4.5.2, “Sample COBOL application to emit the Request event” on page 105.

The transaction must be run four times in quick succession to emulate a customer requesting more than three quotations within the same hour. These transactions then cause the Decision Server Events run time to send the FollowUp action.

Receiving the event

Verify that the Decision Server Events run time received the Request event by logging in to the events administration console at this address:

`http://[hostname]:9088/wbe/common/login.jsp`

Then, follow these steps:

1. Select the Administration view, as shown in Figure 4-32.

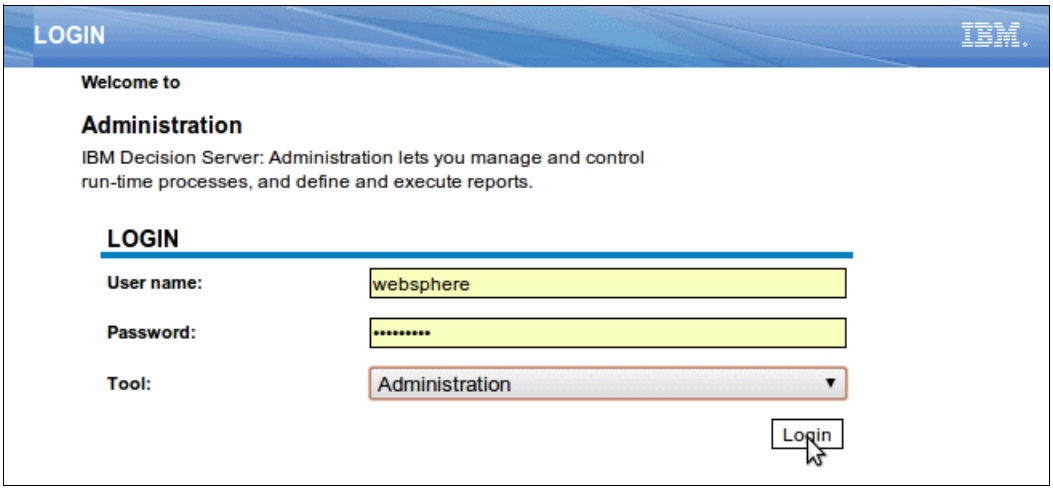


Figure 4-32 Logging in to the events Administration console

2. When logged in, select the reports tool, as shown in Figure 4-33.

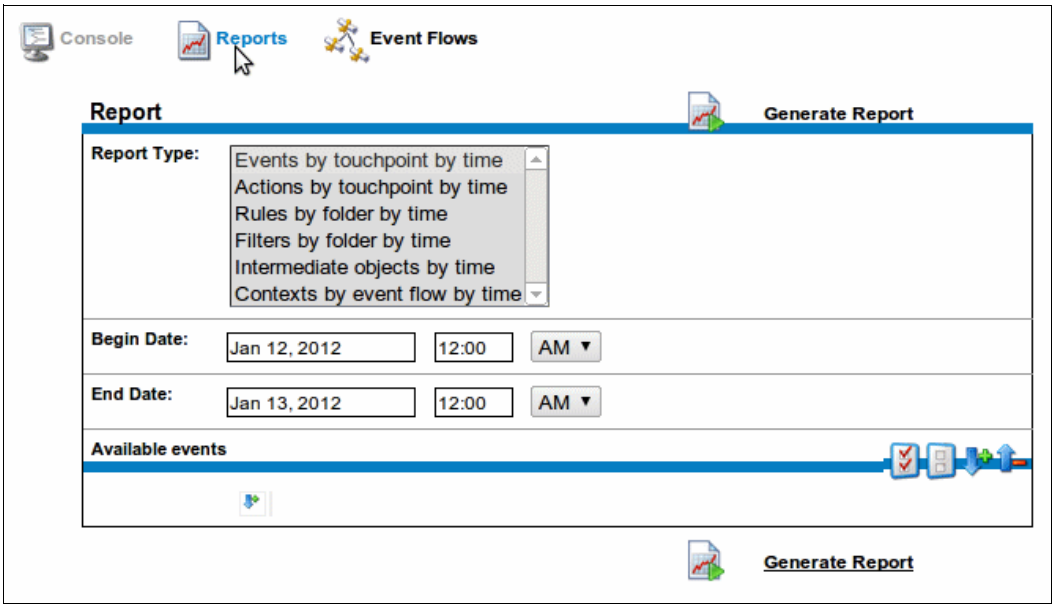
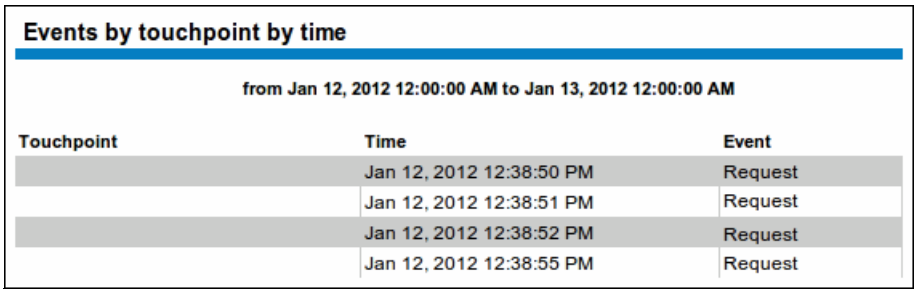


Figure 4-33 Running Generate Reports for the events run time

3. Click **Events by touchpoint by time** and click **Generate Report**. This selection shows all events that are received by the Decision Server Events run time, as shown in Figure 4-34.



Events by touchpoint by time

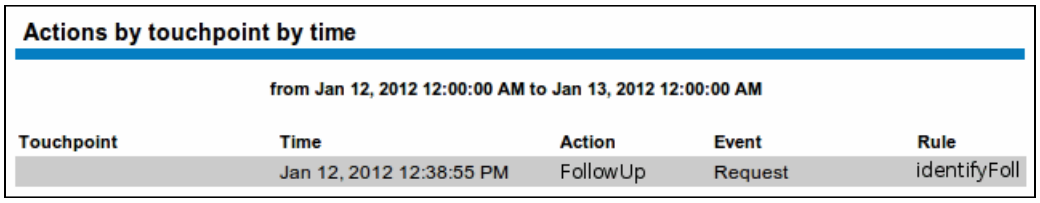
from Jan 12, 2012 12:00:00 AM to Jan 13, 2012 12:00:00 AM

Touchpoint	Time	Event
	Jan 12, 2012 12:38:50 PM	Request
	Jan 12, 2012 12:38:51 PM	Request
	Jan 12, 2012 12:38:52 PM	Request
	Jan 12, 2012 12:38:55 PM	Request

Figure 4-34 The events run time receives the Request event

Checking the action

Because four Request events were received within one hour, a FollowUp action was generated. To check the actions that fired, return to the Reports Tool. Click **Actions by touchpoint by time** and click **Generate Report**. This selection shows all actions fired by the events run time, as shown in Figure 4-35.



Actions by touchpoint by time

from Jan 12, 2012 12:00:00 AM to Jan 13, 2012 12:00:00 AM

Touchpoint	Time	Action	Event	Rule
	Jan 12, 2012 12:38:55 PM	FollowUp	Request	identifyFoll

Figure 4-35 The FollowUp action fired

This step completes the scenario for sending an event from CICS to the Decision Server Events run time. The next section describes various considerations when running the event technology connectors on z/OS.

4.6 Using connectors to receive events from various z/OS sources

Technology connectors are components that provide codeless connections for events and actions by various protocols, including SMTP, HTTP, and FTP. There are two categories of technology connectors: event connectors and action connectors. An *event connector* is used to receive events into the event run time. An *action connector* is used to send an action from the event run time to a target system.

A technology connector is associated with an individual event or action. All the specifications are defined as properties of that event or action in the Event Designer. When a technology connector is defined for an event or action, the icon of the event or action changes to help identify the type of technology connector. Enabling a technology connector requires no coding.

In 4.2.7, “Configuring the technology connectors” on page 87, we describe the full list of connectors available on z/OS. We list the considerations when running these connectors on z/OS. If we did not discuss a connector in this section, it has no special considerations for z/OS. For example, the Email connector uses SMTP to send or receive an email. The Email connector is configured in the same way for z/OS as for any other platform.

4.6.1 Connectors running in the WebSphere Application Server

The event run time on WebSphere Operational Decision Management V7.5 introduces a new type of connector that runs inside WebSphere Application Server. The HTTP, JMS, REST, and SOAP connectors are available as WebSphere applications and are installed to the server where the event run time is installed. The connector applications are installed and updated automatically as required in response to the event run time reloading.

For information about the WebSphere connectors, go to this website:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.admin.doc/doc/runningconnectorsinthewasenvironment.html

JMS queue connector

The JMS queue connector runs as a WebSphere application and can be used to receive events and send actions using WebSphere MQ on z/OS.

In this chapter, we used the HTTP connector to send an event from CICS to the Decision Server event run time. CICS also supports the JMS queue connector, which can be used to send the same event using WebSphere MQ. Using WebSphere MQ increases the reliability of the transport, ensuring that the event is received despite any planned or unplanned interruptions.

4.6.2 Connectors running as a stand-alone batch job

The traditional method for running the technology connectors on z/OS is using a stand-alone batch job that launches the Java process in which the connectors run. Use the WebSphere Application Server connectors where possible because they build on the WebSphere framework. This framework provides high availability and increased manageability of the connectors.

Go to this website for information about the stand-alone connectors:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.admin.doc/doc/zos_runningtechnologyconnectors.html

File system connector

The file system connector can be used to read events from the z/OS Distributed File Service (zFS) and write actions to the zFS. The files contain the Decision Server event XML action packets.

JDBC connector

The JDBC event connector generates events by retrieving data from DB2 tables that are defined as data connections. The JDBC action connector performs basic SQL functions, such as SELECT, INSERT, UPDATE, and DELETE against DB2 tables.

5

Managing business decisions through the full lifecycle

In this chapter, we look at the lifecycle of a decision and considerations when deploying to a z/OS environment.

This chapter contains the following sections:

- What is the lifecycle of rule artifacts in decisions
- Sharing decision artifacts between z/OS and distributed
- Installation topologies for Decision Center

© Copyright IBM Corp. 2012. All rights reserved.

5.1 What is the lifecycle of rule artifacts in decisions

One of the key reasons for extracting decisions into a decision management system is that it allows you to separate the lifecycle of the decisions from the lifecycle of the application that invokes the decision. This separation is important particularly on System z where often the application deployment cycles are long. And, the business wants to be able to change the business behavior of the application more quickly. By separating the decision lifecycle from the application lifecycle, we can gain agility in our business applications without sacrificing the reliability of the core logic of our business applications.

The decision lifecycle starts with the initial location of the rules within the business applications. After the rules are identified, the data that is associated with the decisions can be identified and a copybook can be created that contains the required information.

The business application must be refactored to remove the current implementation of the business decision and replace it with calls out to the zRule Execution Server for z/OS. The application might have decision logic and business logic interspersed throughout it. To optimize the application, refactor the application so that it makes the fewest calls possible to decisions, as shown in Figure 5-1.

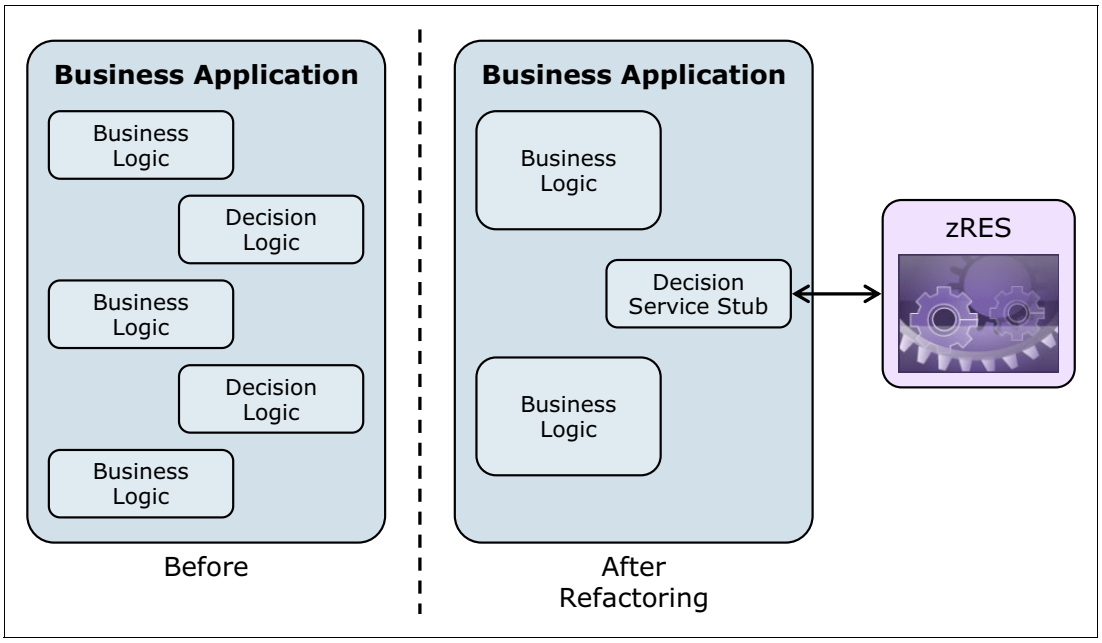


Figure 5-1 Refactoring the business application

After the copybook containing the required data for the decision is created, the business rules for the initial version of the business decision can be authored. The initial creation of the business object model (BOM) and verbalization is done in Rule Designer. This action is generally considered an IT project, because it requires knowledge of the COBOL data structure. After the BOM is created, the IT department typically creates the initial version of the business rules for the decision based on the rules that were previously embedded in the application. After the first version of the business rules is created, the decision can be deployed to a zRule Execution Server environment and tested with the application.

With the first version of the decision now deployed, you must decide how to manage the ongoing lifecycle of the decision. There are multiple approaches to this problem. In certain development shops, the management of the business decisions remains a purely IT-based

process, using the Rule Designer environment for managing and maintaining the decisions. Now that the business rules from the decision are authored in a far more accessible language, it is often advantageous to allow the business team to interact directly with the authored rules. This approach uses the Decision Center environment. Figure 5-2 shows a high-level interaction between the decision components and users.

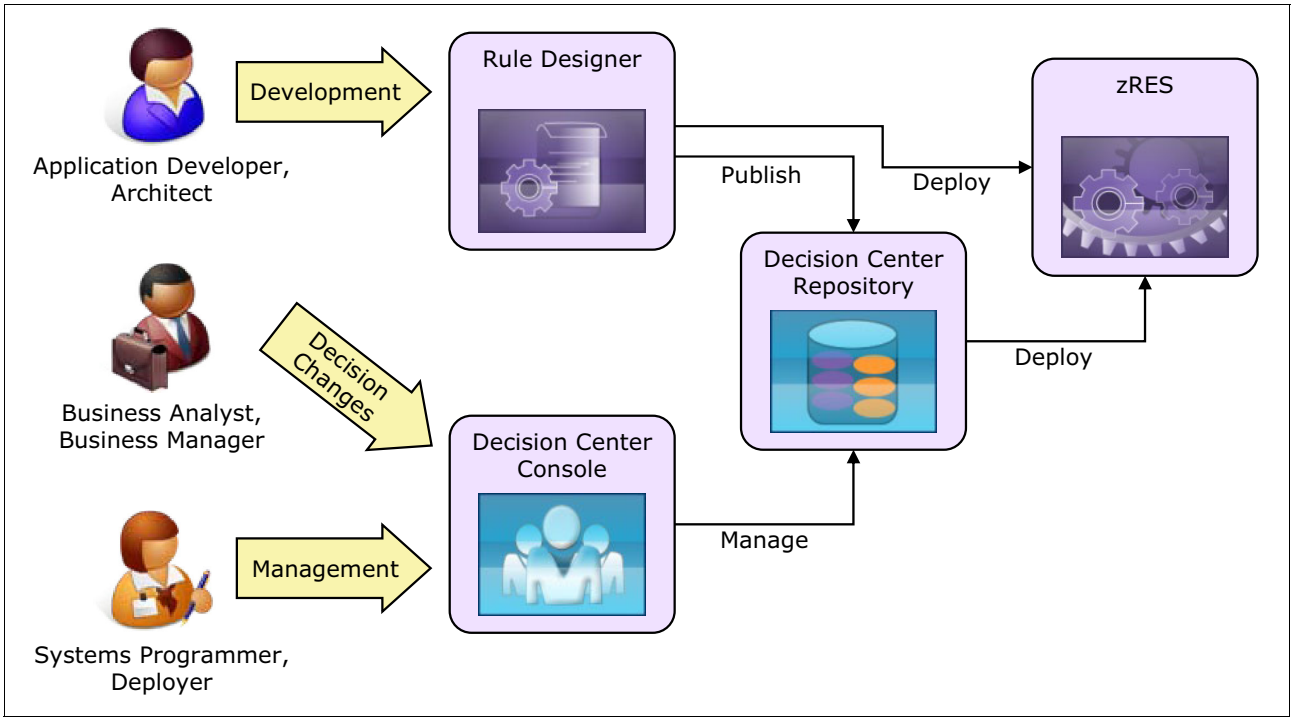


Figure 5-2 Users and the rule artifacts

After the initial deployment of the rules, the application developer can also publish the rules to the Decision Center repository. Other roles can become involved in the decision lifecycle. The Decision Center console provides two distinct functions. One function is the ability to change the rules of decisions that are published to the repository. Another function is the ability to manage versions and deploy decisions from the repository to the rule runtime environments.

An important part of the decision lifecycle is the ability to test the changes to the decision before deploying it to the final server. You can use Decision Center to define scenarios or test cases using, for example, a spreadsheet format to define the input and expected output. Decision Center can then take this data, deploy the ruleset to a configured Rule Execution Server, and execute the decisions based on the supplied data. Decision Center can report situations where expected results are not returned. This function gives you the ability for the decision lifecycle to happen completely in isolation from the application lifecycle. Changes to the decision behavior can be tested in isolation from the application before they are deployed into the production system. We explain this topic in detail in Chapter 6, “Decision testing and simulation” on page 121.

5.1.1 Managing artifacts

There are a number of artifacts in the decision lifecycle that require managing. We list them here, and their interactions are shown in Figure 5-3 on page 114:

- Ruleset
- Java execution object module (XOM)

- Marshaller project
- RuleApp

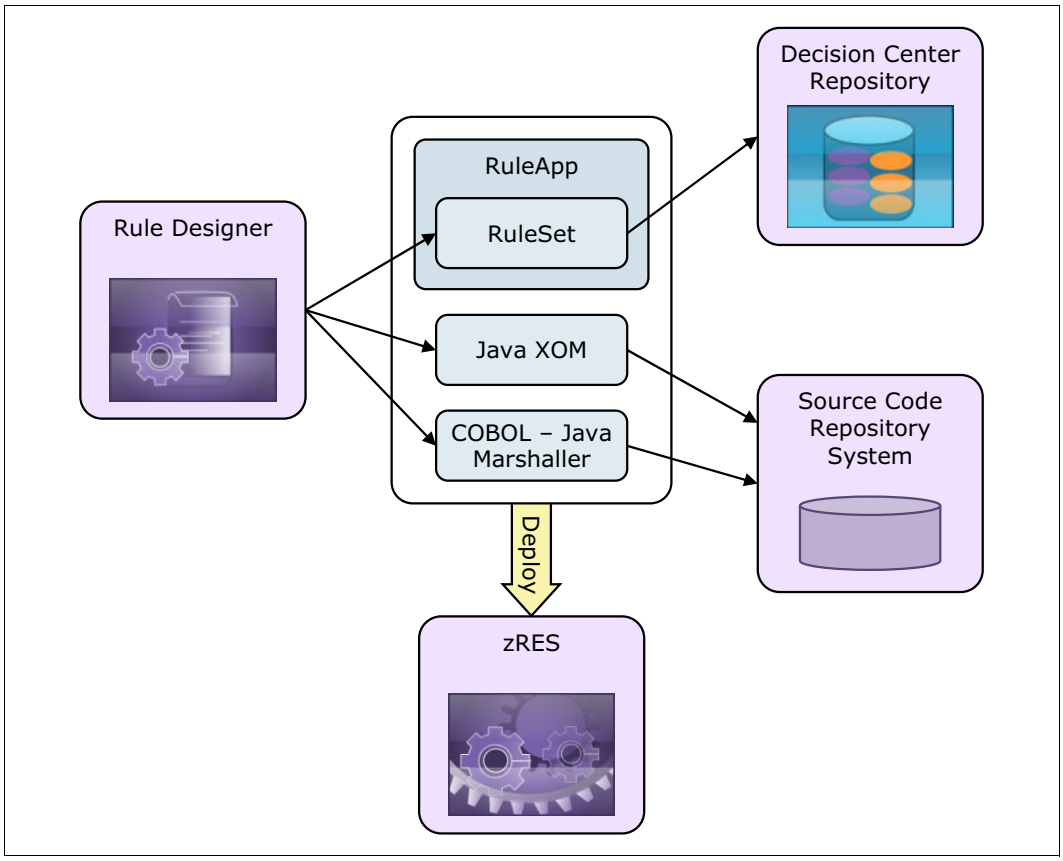


Figure 5-3 The artifacts within the decision lifecycle

Ruleset

The *ruleset* contains a number of elements:

- The business object model
- The verbalization of that model
- The authored rules from the decision
- The ruleflow that guides the execution of the decision
- The declaration of the required parameters for this decision

The ruleset is the primary artifact that requires management. It contains the rules themselves and the business object model on which they are based. The *ruleset* is the artifact that is published to Decision Center and you can access the Decision Center to change the rules within a decision.

A ruleset can inherit from another ruleset. In this case, the decision contains the BOM, verbalization, and rules from both rulesets. This approach is a useful way to reuse rules that are shared across multiple decisions and manage the changes to the shared rules in a single project. The name of a ruleset is significant. It forms part of the ruleset path that is used by the client to identify the decision that it wants to invoke on the server. Any changes to the rules within a decision require the ruleset to be redeployed to make the new decision behavior available. A preferred practice is to increment the minor version of the decision when redeploying behavioral changes in the decision.

Java XOM

The *Java XOM* is standard Java code that is the Java representation of the imported COBOL copybook. It is required at run time in the zRule Execution Server for z/OS to map the incoming COBOL data structure into before the rules are executed. The Java XOM can be deployed to the zRule Execution Server for z/OS server directly from Rule Designer or via scripts on the server instance.

Because the code is standard Java code, it must be managed and maintained using a source code management system. Rule Designer is based on Eclipse. Many of the standard source code management systems have plug-ins that allow you to manage and handle versioning for the Eclipse Java project directly from the Rule Designer environment. If the starting point for the project is an imported copybook, this code is generated by Rule Designer and must not be edited. If possible, mark this code as read only within the source code management system.

Only deploy the Java XOM if there are changes to the underlying data structures that define the interface to this decision. This situation occurs when a change is made to the COBOL copybook that was imported to create the business object model. A change of this nature also requires a corresponding change in the COBOL client applications to use the new copybook structure. For this reason, changes to the COBOL copybook are considered an IT project and happen less frequently than changes to the decision behavior in the rules. A preferred practice is to increment the major version number of the decision when making these interface changes.

Marshaller code

The marshaller code is standard Java code that is used to transform data between COBOL and Java. It is required at run time in the zRule Execution Server for z/OS to perform the mapping of the incoming COBOL data structure into the Java XOM before the rules are executed. The marshaller code can be deployed to the zRule Execution Server for z/OS server directly from Rule Studio or via scripts on the server instance.

Because the code is standard Java code, it must be managed and maintained using a source code management system. Rule Designer is based on Eclipse. Many of the standard source code management systems have plug-ins that allow you to manage and handle the versioning of the Eclipse Java project directly from the Rule Designer environment. If the starting point for the project is an imported copybook, this code is generated by Rule Designer and must not be edited. If possible, mark this code as read only within the source code management system.

Only deploy the marshaller if changes are made to the underlying data structures that define the interface to this decision. This situation occurs when a change is made to the COBOL copybook that was imported to create the business object model. A change of this nature also requires a corresponding change in the COBOL client applications to use the new copybook structure. For this reason, changes to the COBOL copybook must be considered an IT project and happen less frequently than changes to the decision behavior in the rules. A preferred practice is to increment the major version number of the decision when making these interface changes.

RuleApp

The *RuleApp* is the deployment container for one or a number of related rulesets. RuleApps are created for deployment either within Rule Designer or Decision Center. They are basically a zip file that contains the required artifacts to execute the decision.

A RuleApp can be deployed directly to a server from either Rule Designer or Decision Center. Or, a RuleApp can be exported as a jar file that can be managed externally to the WebSphere Operational Decision Management tooling and deployed to a server using scripts. This

approach can be useful when defining the process of performing decision updates where there is no access from a Rule Designer or Decision Center to the production zRule Execution Server for z/OS.

The name of a RuleApp is significant. The name forms part of the ruleset path that is used by the client to identify the decision that it wants to invoke on the server.

5.1.2 What roles are involved in the decision lifecycle

There are three roles that are involved in the lifecycle of a decision. The names vary from company to company, but there are normally people who can be attributed to one or more of the following roles:

- Application/decision developer
- Systems administrator/programmer
- Business team member

The major interaction between the team members occurs in the Decision Center environment. Here, a developer synchronizes the RuleApps on which the developer is working in Rule Designer. The systems administrator goes to Rule Designer to version and deploy decisions. The business team accesses Rule Designer to view or change decisions.

The Decision Center environment provides role-based access authorities to allow the systems administrator to give people the correct authority to access the rulesets for which they are responsible. The granularity of access authority is delivered at the ruleset level. But, with a little customization, the granularity of management can be changed to match whatever is required by the organization.

Rule developer

The rule developer is generally an IT-based person. In most organizations, the rule developer is also part of the application development team that is responsible for the application that is being modernized by having its decisions extracted. In a larger organization, a dedicated team with the skill to externalize and develop decisions might exist, outside of the application development team.

The rule developer is responsible for creating the initial version of the business object model and the verbalization of that model. The rule developer normally writes the first draft or pass at the rules within the decision based on what currently exists within the application code. The rule developer's primary tool is Rule Designer. The rule developer is responsible for publishing the ruleset to Decision Center. This person also ensures that the Java XOM and marshaller code are maintained in a source code management system.

Systems administrator

The systems administrator is responsible for the production zRule Execution Server for z/OS servers and their actions. In the rule lifecycle, the systems administrator generally is responsible for versioning and deploying new versions of a decision into the zRule Execution Server for z/OS. The systems administrator is also responsible for maintaining the security model within the Decision Center environment to ensure that users have access to only the rulesets for which they are responsible.

Business team

The business team is ultimately responsible for the business policies, which are enforced through the business decisions. The business team provides the input to the behavior of the business decisions. Depending on the level of adoption, the business team's interaction varies.

The business team provides input to the rule development team for business policy changes to implement in the decisions. The business team views the rules from existing decisions in Decision Center and recommends to the rule development team updates to specific rules that must be made to implement the new business policies. In this mode, the business team has read-only access to the rulesets that contain the applicable rules.

The business team changes the rules within a decision directly to implement changes that are required as business policies are updated. The business team tests the changes to the decisions using the testing and simulation capabilities that are accessed through Decision Center. The business team then notifies the systems administrator that a new version of a decision is available and needs to be deployed.

5.2 Sharing decision artifacts between z/OS and distributed

When considering sharing decisions between z/OS and other platforms, ensure that the correct decision artifacts are available on each platform. As part of the decision lifecycle, the required artifacts can come from separate management systems.

Figure 5-4 shows the artifacts that are involved in a decision and the artifacts that are required to be deployed to various platforms.

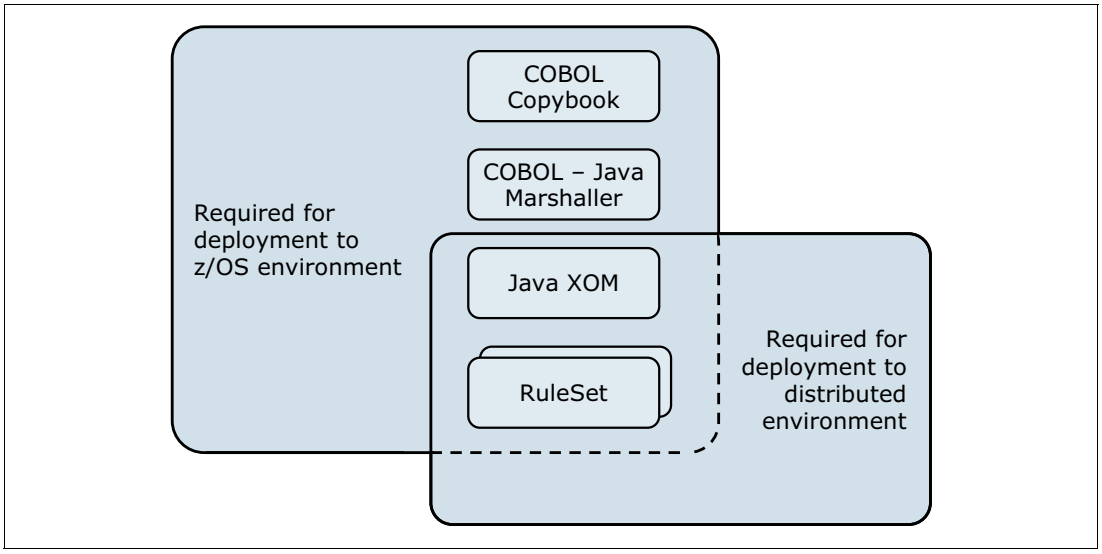


Figure 5-4 Deployment of decision artifacts

Normally, only the rulesets and the Java XOM are required on the distributed platform. Often in this configuration, the client is local to the Rule Execution Server. The Java XOM is part of the client application class path, so it is not explicitly deployed to the server as a resource. The rule session inherits the class path of the client application.

On zRule Execution Server for z/OS, the COBOL copybook that was either imported into, or generated by, Rule Designer is required by the client COBOL application. The COBOL copybook ensures that the data layout is exactly what is expected by the marshaling code. Because the client in this case is COBOL and the connection to the server is managed by the zRule Execution Server for z/OS API stub, the Java XOM and marshaller resources must be deployed to the server. This deployment can be done either from the Rule Designer environment or locally using supplied scripts.

In both cases, the Rule Execution Server requires that the ruleset is deployed within a RuleApp. This deployment is from either Rule Designer, Decision Center, or locally using scripts.

When sharing the decision across multiple platforms, it is important to make sure that the decision lifecycle updates and deploys the correct parts of the decision when changes are made. If the underlying data structure definitions are not changed when updating a decision, only the rulesets within a RuleApp must be redeployed to make the new decision version available. If changes are made to the copybook used to create the business object model, or if the Java XOM was used to create the copybook, you need to redeploy artifacts. Redeploy artifacts on all platforms where the decision is implemented to minimize the chances of unexpected behavior or failures. For this reason, changes to the data model must be minimized after the decisions are in production and considered an IT project to implement.

5.3 Installation topologies for Decision Center

The locations of the Decision Center repository and the console are largely independent of where the decision executes. Figure 5-5 shows the possible options for installing a Decision Center to be used with zRule Execution Server for z/OS servers on z/OS.

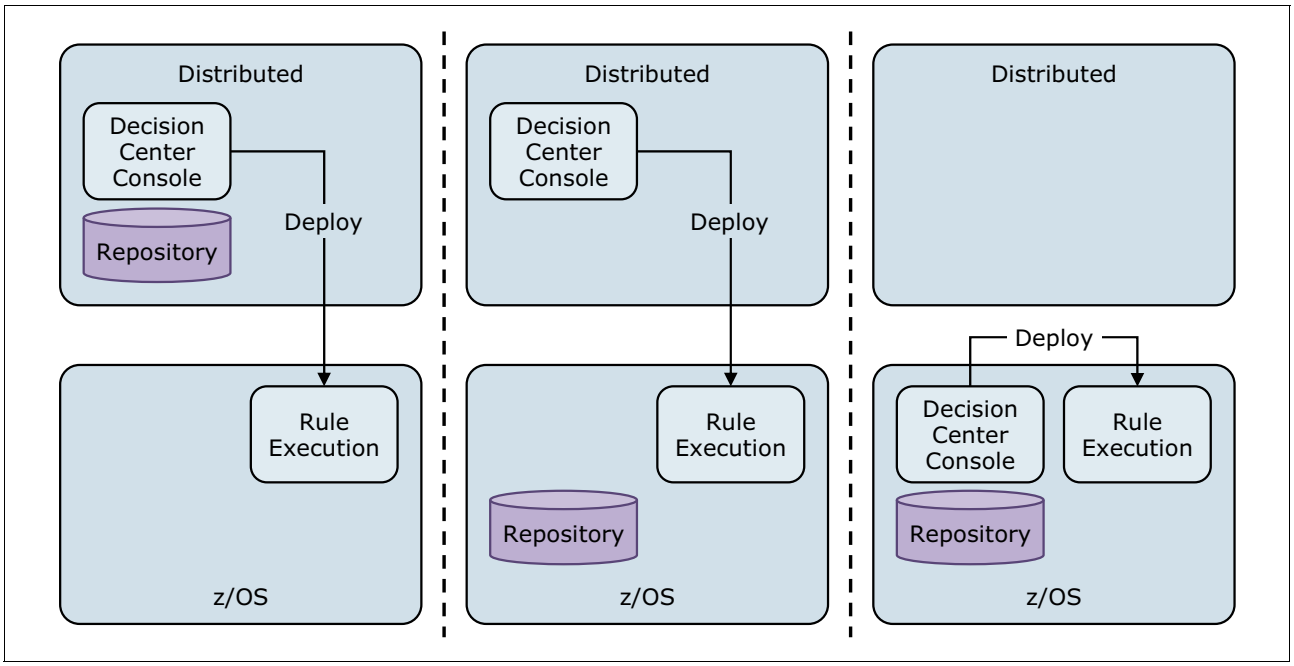


Figure 5-5 Deployment options for Decision Center repository and console

The Decision Center console requires deployment to a web container. This web container can be any one of the supported Java Platform Enterprise Edition (Java EE) application servers for distributed or WebSphere Application Server for z/OS. The Decision Center console also requires Java Database Connectivity (JDBC) access to the repository database.

Topology 1: Decision Center console and repository on distributed

In this topology, both the Decision Center console and repository are hosted on a distributed or Linux for System z platform. The standard deployment from the Decision Center console to a Rule Execution Server is by HTTP-based communication. As long as access is granted to the specific ports that are configured for deployment on the zRule Execution Server for z/OS

or other RES instances, it is straightforward to deploy to any RES/zRule Execution Server for z/OS instance. To Decision Center, the deployment interface to zRule Execution Server for z/OS looks the same as other Java EE-deployed RES instances.

All security to the console is role-based using the underlying application server to perform authentication checks. You might prefer this configuration, because it does not require all users of the Decision Center console to be defined to z/OS security.

Topology 2: Decision Center console on distributed and repository on z/OS

This topology is similar topology 1; however, the database is on a z/OS logical partition (LPAR). In this case, the Decision Center console must also have ports that are enabled in any firewall to allow remote client access to the database on z/OS. Generally, Decision Center is used less than Decision Server, so the remote location of the database is not a performance problem.

You might prefer this configuration if the rule repository is required to have z/OS qualities of services (QoS) associated with it or if the rule repository is managed on the same platform as the COBOL source code repository. Generally, in this case, the security access to the remote database is delegated to an application server-level connection. That way, each Decision Center console user does not have to be defined to z/OS security.

Topology 3: Decision Center console and repository on z/OS

This topology places both the Decision Center console and repository on z/OS. The Decision Center console requires a WebSphere Application Server for z/OS instance in which to run and the repository requires a DB2 instance. This topology can also be deployed to distributed RES instances, as well as to zRule Execution Server for z/OS instances.

You might prefer this configuration if all administration and deployment of a project are contained within the z/OS teams.

6

Decision testing and simulation

This chapter describes how to run tests and simulations for your solution. It contains the following sections:

- Making the right testing and simulation decisions
- Testing and simulation architecture for z/OS decision services
- Testing and simulation lifecycle
- Running tests and simulations from Rule Designer and Decision Center

© Copyright IBM Corp. 2012. All rights reserved.

6.1 Making the right testing and simulation decisions

There are two critical aspects of all software development projects. One critical area is testing the business logic before deploying it into production. The other critical area is optimizing the decisions by running what-if analyses on a set of reference data. You run the analyses to evaluate the effect of a business logic modification if deployed into production.

Developers might use their traditional development toolset to run tests and simulations against a decision service. However, business users generally do not have access to an offering. Implementing and deploying a custom solution to provide them with these essential services can generate extra costs and delays to the project.

Because the business users are involved in the definition and the deployment of decision services developed with WebSphere Operational Decision Management for z/OS, dedicated testing and simulation tools are provided with the product:

- After a rule project is deployed into Decision Center, business users can use Microsoft Excel to define a set of tests or simulation scenarios. They can save them and version them as artifacts of the rule project, run them against any version of the ruleset or any subset of its rules, and then access a detailed execution report to analyze the results of the execution.
- Various levels of detail are available in the execution reports. A business user can generate a Microsoft Excel file that contains all the inputs and outputs of the decision service under test. The business user can reuse the file to generate custom reports using familiar business intelligence tools.
- Because the same tests and simulation scenarios can be run against various versions of the same ruleset, business users can compare two or more execution reports to spot the differences between them.
- To accommodate special needs in terms of tests and simulation data sources, a public Java API is available to author custom formats of tests and simulations and deploy them into Decision Center to make them available to business users. Rule Designer provides a set of dedicated editors and a wizard to streamline both the development and the deployment of a customization. A common use case for customization is the deployment of a custom scenario format that can run simulations using reference data stored in an enterprise database or a set of pre-existing reference files.

WebSphere Operational Decision Management for z/OS uses the term *scenario suite* to refer to a set of test cases or a set of simulation cases that can be run against a ruleset. A scenario suite is defined as a set of scenario cases. Each scenario defines the value of the decision service inputs, and optionally a set of expectations about the output (if the scenario suite is a test suite).

6.1.1 Verifying the business logic implementation by testing

The first type of scenario suite that can be created with WebSphere Operational Decision Management for z/OS is the test suite, which defines a list of tests to run against a ruleset.

Each test suite is defined as a list of test cases, and each test case is defined by these characteristics:

- Its name
- The value of the input parameters to submit to the decision service

- An optional list of expectations about the output parameters of the decision service (“the price of the proposal equals …”. “the reward level of the customer is …”. and so on)
- An optional list of expectations about the execution details

Execution details provide detailed information about what is happening in the rule engine during the execution of a ruleset. When the user who defines a test suite also implements the ruleset, the user can define tests on the following execution details:

- The list of rules that were fired during the execution of the ruleset
- The list of rules that were not fired during the execution of the ruleset
- The total number of rules that were fired during the execution of the ruleset
- The total number of rules that were not fired during the execution of the ruleset
- The list of tasks that were executed during the execution of the ruleset
- The list of tasks that were not executed during the execution of the ruleset
- The total number of tasks that were executed during the execution of the ruleset
- The total number of tasks that were not executed during the execution of the ruleset
- The duration of the ruleset execution, for basic performance testing

This way, the decision service author can test, for example, if certain values for the input parameters trigger the execution of certain rules.

Initial testing

The first test suites that are generally run against a decision service are the tests that verify that the decision service behaves as expected prior to its deployment to production.

WebSphere Operational Decision Management test suites support the creation of a test suite, as soon as the signature of the decision service is defined, to support test-driven development. WebSphere Operational Decision Management test suites support the creation of test suites with initially empty expectations. These expectations are later specified using copy and paste between an execution report resulting from a first “dry” run of the test suite against the ruleset and the test suite definition in Microsoft Excel.

Regression testing

After successful initial testing, a decision service can be deployed into production. Subsequent changes to the decision service logic generally imply that you perform regression testing on the updated decision service, before deploying it into production. This way, you can ensure that the new business logic does not cause unwanted side effects.

By providing versioning of the test suites and an execution reports comparison feature, WebSphere Operational Decision Management for z/OS allows its users to easily run regression testing campaigns before deploying an update of the business logic of a decision service.

6.1.2 Simulation: Running what-if analysis and fine-tuning the decision logic

Simulation is the second type of scenario suite that you can create with WebSphere Operational Decision Management. WebSphere Operational Decision Management defines a list of simulation cases to run against a decision service and a list of key performance indicators (KPIs) to be calculated during the execution.

A simulation case is defined by these characteristics:

- Its name
- The values of the input parameters to submit to the decision service

The KPIs are defined at the scenario-suite format level and are not defined by the business user.

In the simulation report, the values of all calculated KPIs are displayed. The same simulation can be run against separate versions of the same ruleset. Separate reports can be compared to check the differences between the KPI values. This capability allows the business user to fine-tune the content of the business logic between various executions of the same simulation until the desired KPI values are attained.

6.2 Testing and simulation architecture for z/OS decision services

This section discusses the testing and simulation architecture for decision services on z/OS.

6.2.1 Test and simulation artifacts

This section lists the test and simulation artifacts.

Scenario suites

A scenario suite is an artifact that is created in Decision Center to define either a test suite or a simulation (Figure 6-1).

A scenario suite is attached to a rule project (or a ruleset), for which it defines tests or simulation cases. After its creation, you can run a scenario suite from within Decision Center at any time. Each execution creates an execution report that is stored along with the scenario suite in the Decision Center database.

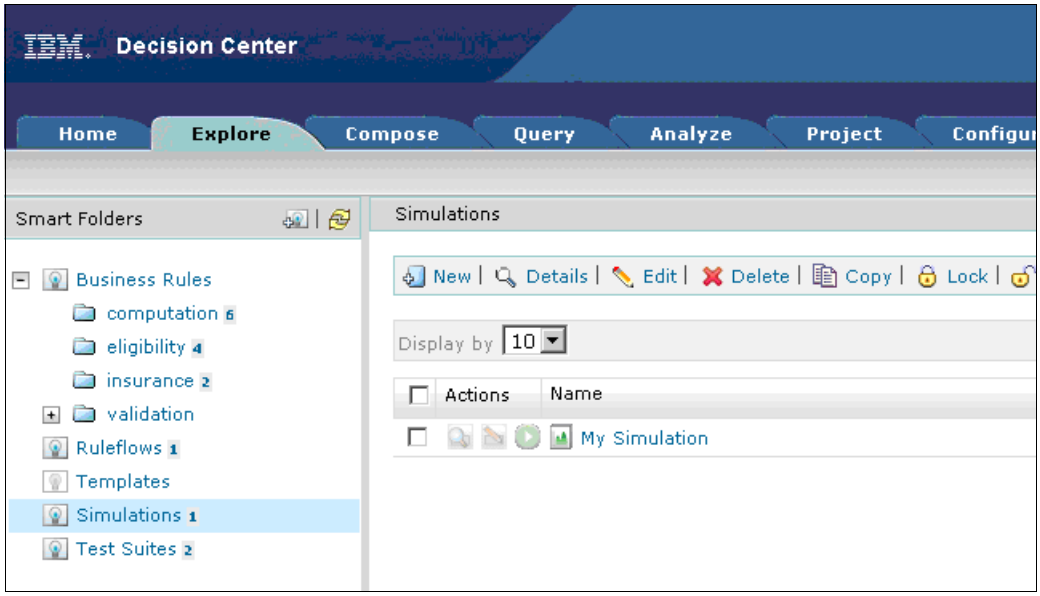


Figure 6-1 Scenario suite artifacts in Decision Center: Simulations and test suites

Decision Validation Services archives

A Decision Validation Service (DVS) archive is an artifact that contains both a scenario suite and its associated ruleset in a single file that is easy to share. Scenario suite archives can be exported from Decision Center and imported into Rule Designer, for execution and debugging

by a developer. This archive is used when errors occur in the ruleset during the execution of a scenario suite, and the business user needs help from a support team to understand what happened.

Scenario suite formats

When creating a test suite or a simulation in Decision Center, the business user is presented with a list of supported formats by the scenario suite creation wizard (Figure 6-2). These formats are the scenario suite formats. They encapsulate the information that is required by the testing and simulation features to understand how a scenario suite is defined and needs to run.

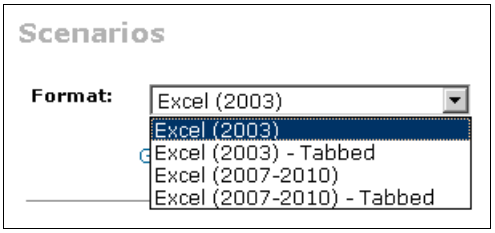


Figure 6-2 Selection of a scenario suite format when creating a scenario suite in Decision Center

The scenario suite format defines the following information:

- The scenario provider implementation to use at run time. The scenario provider is a Java class that implements the ilog.rules.dvs.core.IlrScenarioProvider interface. The scenario provider performs the following actions:
  - The scenario provider retrieves scenario data and returns a set of input parameter values for each scenario case.
  - If the scenario is a test suite, the scenario provider provides the set of tests to be run on each scenario case execution result.
  - The scenario provider typically retrieves the scenario data and the test specification from a dedicated data store (a relational database or a file, for example) that is provided at creation time by the user running a test suite or simulation. Testing and simulation for decision services on z/OS come with a predefined scenario provider. This scenario provider retrieves both the scenario data and test specifications from a Microsoft Excel workbook that was previously generated for a ruleset in Decision Center or in Rule Designer. This scenario provider is referred to as the *Excel scenario provider*. Information about the data store to use for a scenario provider is retrieved from a dedicated graphical user interface in Decision Center and Rule Designer. For Decision Center, this user interface is defined by a scenario suite resource renderer class. This scenario suite resource renderer class is a Java class that implements the ilog.rules.teamserver.web.components.renderers.IlrScenarioSuiteResourcesRenderer interface. For Rule Designer, this user interface is defined by an Eclipse launch configuration plug-in.
- Both the rendered class and the launch configuration plug-in fully qualified names are stored in the scenario suite format, along with the scenario provider implementation fully qualified name.
- The precision (number of meaningful digits after decimal point) to use when testing numbers. For example, you might want number comparison to be only meaningful up to two digits after the decimal point; consider that an observed value of 2.55457 matches an expected value of 2.55.
- The list of execution details that can be tested when using this format to create a test suite.

- The type of ruleset parameter values to be returned in an Excel file along with an execution report when the user requests it in Decision Center. Either the input parameter values or output parameter values can be returned.
- The list of KPIs to be calculated when a simulation is run using this format.

The scenario suite formats are created using dedicated wizards and editors in Rule Designer. They are then published in Decision Center to be viewed by the business user. After a scenario suite format is published in Decision Center, it needs to be enabled on a per project basis by a Decision Center administrator. Then, a business user can use the scenario suite format to create a test suite or simulation. This way, a custom scenario suite format with a list of dedicated custom KPIs that you define for running a simulation on a particular rule project is not visible in other projects.

By default, the following scenario suite formats are available in Decision Center after a standard installation of the product (Figure 6-3):

- Excel (2003) format: Using this format, users can define a test suite or simulation using a Microsoft Excel 2003 workbook. One page is dedicated to the definition of the scenario case. Each input parameter is defined in a table, one column per attribute, and the tests are defined in dedicated pages.
- Excel (2003) - Tabbed format: This format, which is not enabled by default, is similar to the Excel (2003) format, except that the user can use more than one page to define values for the scenario cases to describe complex object structures.
- Excel (2007-2010) format: This format, which is not enabled by default in Decision Center, is similar to the Excel (2003) format, except that it uses a workbook that is compatible with Microsoft Excel 2007 and 2010.
- Excel (2007-2010) - Tabbed format: This format, which is not enabled by default in Decision Center, is similar to the Excel (2003) tabbed format, except that it uses a workbook that is compatible with Microsoft Excel 2007 and 2010.

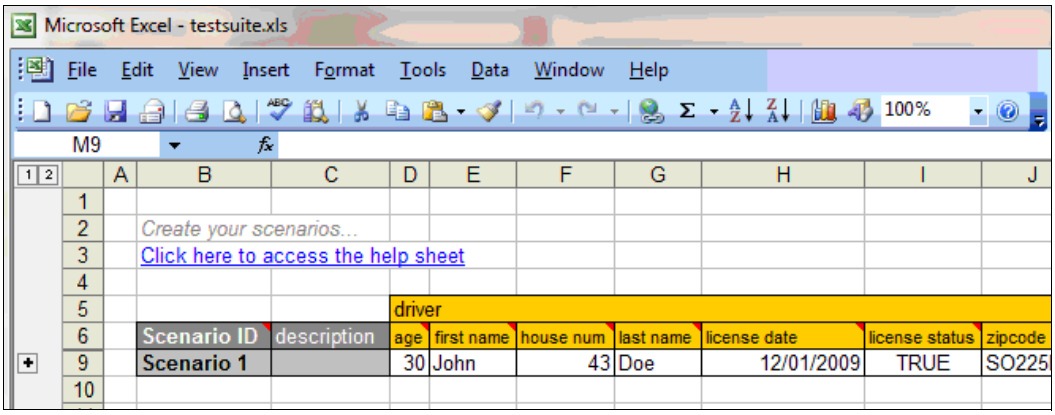


Figure 6-3 Excerpt from an Excel scenario suite created with one of the Excel formats

Extensive documentation on the Excel format is provided in the WebSphere Operational Decision Management documentation center:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/index.jsp?topic=%2Fcom.ibm.dcenter.bu.rules%2FContent%2FBusiness_Rules%2F_pubskel%2FInfocenter_Primary%2Fps_DCBU_DCenter_Rules2495.html

All Excel formats are suitable for running test suites initially, but they do not define any KPIs. The KPIs are specific to a rule project. You must customize the KPIs for these formats prior to using the formats for simulations (Figure 6-4 on page 127).

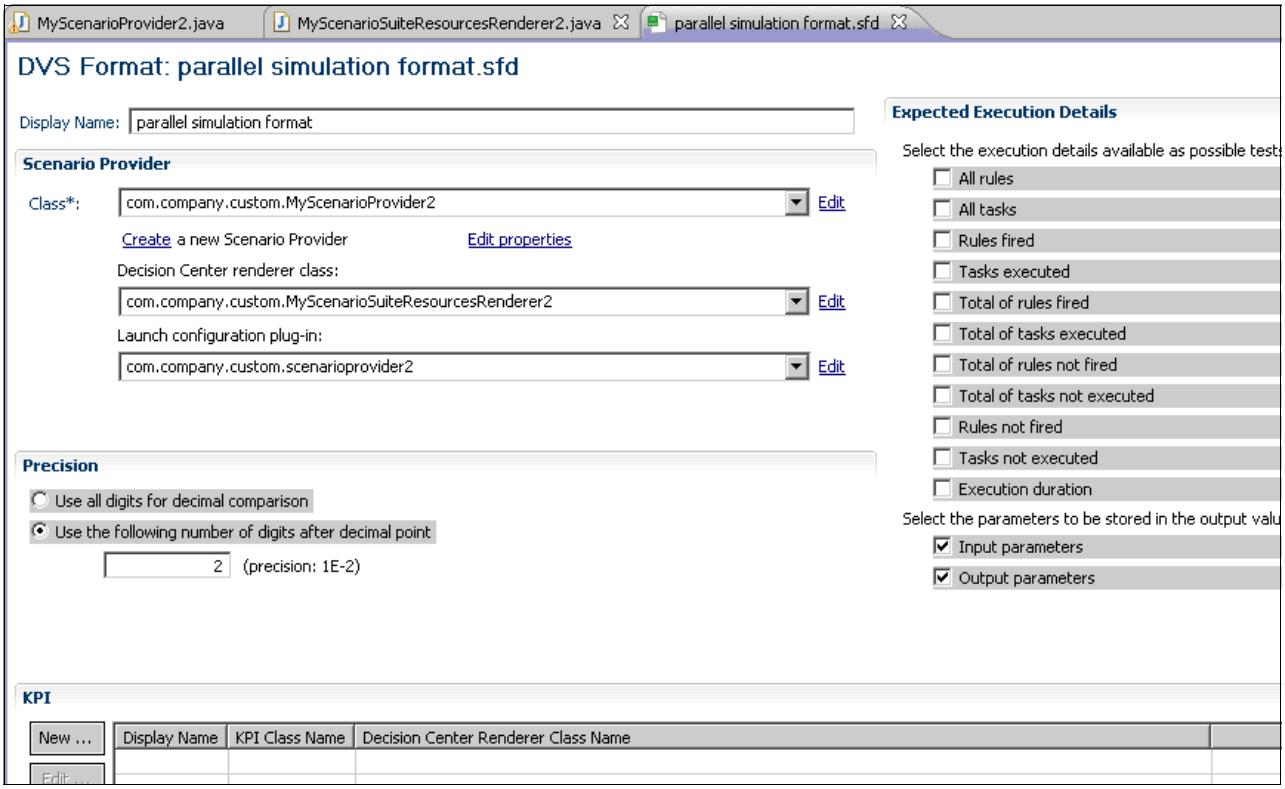


Figure 6-4 The Scenario Suite Format Editor in Rule Designer

DVS launch configurations

In Rule Designer, you can run test suites only (simulations can be run from Decision Center only). Two types of Eclipse launch configurations are provided to run test suites:

- The DVS Excel File launch configuration is provided so that you can run a test suite defined in a Microsoft Excel Workbook locally or against a remote server (Figure 6-5 on page 128).
- The DVS Archive launch configuration is provided so that you can run a test suite defined in a DVS archive retrieved from Decision Center locally or against a remote server.

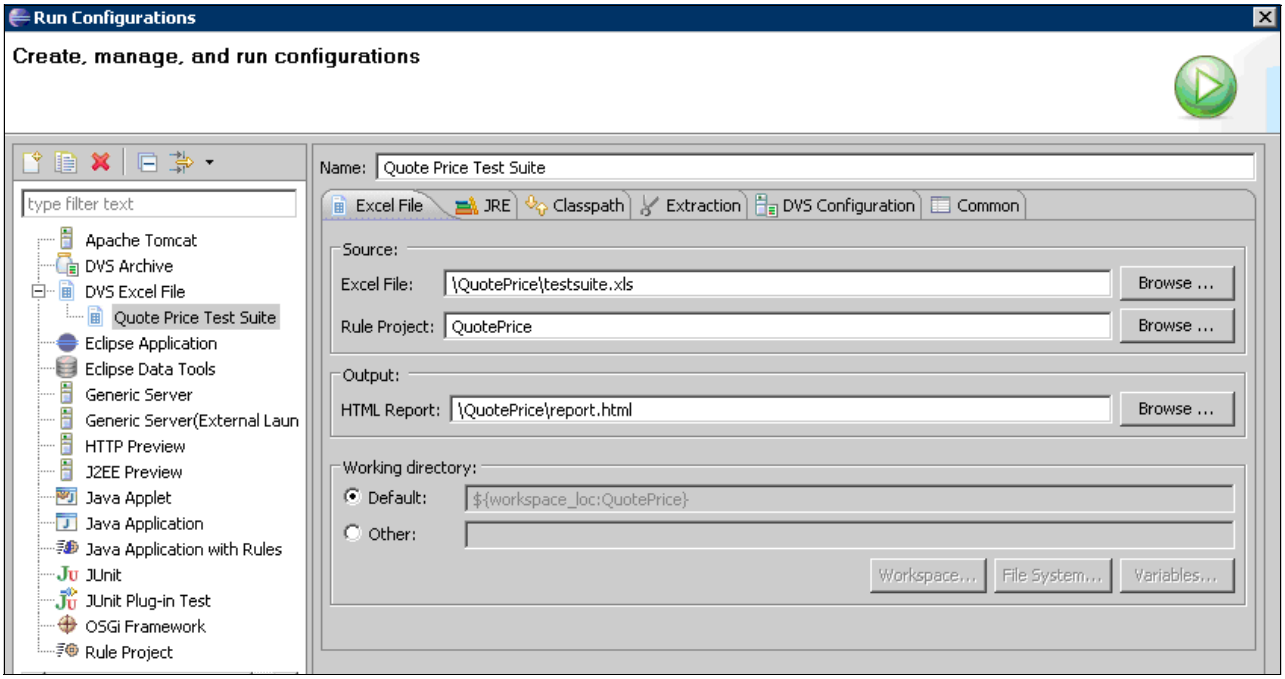


Figure 6-5 A launch configuration for an Excel test suite in Rule Designer

In addition, you can create custom launch configurations using the Eclipse plug-in extension to run test suites, which are defined using a custom scenario suite format. They are not based on the Excel scenario provider.

Test and simulation reports

Two types of test and simulation execution reports are provided. One type is for developers running test suites from Rule Designer. The more extensive type is for business users running test suites or simulation from Decision Center.

Execution reports for developers

The test suite execution report for developers is generated as an HTML file in the Rule Designer workspace after the execution of a test suite (Figure 6-6 on page 129). It includes the following information:

- The date and time that the test suite was run
- The type of execution (local or remote)
- The number of digits after the decimal point that is used for number comparisons
- The total number of scenarios in the test suite
- The total number of tests in the test suite
- The success rate for the test suite (percentage of test cases that were successfully verified)
- The total number of test failures
- The total number of test errors
- For each scenario, the name of the scenario, the success rate for the scenario, the number of tests in the scenario, the number of failures, and the number of errors
- For each test of a scenario, the name of the test, the result of the test execution (success, failure, or error), and a message that explains the result of the test execution

WebSphere. Execution Report

December 2, 2011 1:07:29 PM GMT

IBM.

Summary

Execution	Local
Decimal Precision	2 digits are used
Scenarios	1
Tests	4
Success Rate	75%
Failures	1
Errors	0

Details for all Scenarios

Name	Success Rate	Tests	Failures	Errors	Message
Scenario 1	75%	4	1	0	

Details by Scenario

Scenario 1

Name	Success	Failure	Error	Message
the response is approved equals	√			The observed value false is the expected value.
the base price of the response equals	√			The observed value 2000.0 is the expected value.
the dis price of the response equals		x		The observed value **1850.0** does not equal the expected value **1800.0**
the list of fired rules contains	√			The set of values [MinimumAge] contains the expected values [MinimumAge].

Figure 6-6 A test suite report in Rule Designer

Execution reports for business users

In Decision Center, an execution report is generated internally and stored in the Decision Center database along with the scenario suite information every time that a scenario suite is run. A report can be retrieved at anytime from the scenario suite artifact in the Decision Center explorer. Various reports for separate versions of the same ruleset can be compared to help you detect regressions.

When running a scenario suite from Decision Center, many options are available to customize the content of the scenario suite execution report (Figure 6-7).

Run My Test Suite - Version: 1.0

Server: Local server

Report Options:

Name My Test Suite - Repor

☑ Create an Excel file separate from the report to store the output values

In addition to any tests specified in the scenario file, you can add in the report:

☑ The list of rules fired
☑ The list of executed ruleflow tasks
☑ The duration (in ms) of execution

Cancel Run

Figure 6-7 Detailed reporting options for a test suite in Decision Center

The additional reporting options are useful to analyze the results of an execution in detail. However, you must use them sparingly, because they have a large impact on the performance of the runtime execution.

A scenario suite execution report for business users displays the following information:

- The name of the report.
- The version of the test suite or the simulation for which the report was generated.
- If the scenario suite uses the Excel scenario provider, a link is provided in the report to download the Excel workbook.
- The date and time that the scenario suite was run.
- The name of the user that launched the scenario suite.
- The rule selector that was applied to the set of rules for running the scenario suite, and the baseline that was used.
- The starting ruleflow task for the execution.
- The testing and simulation server against which the scenario suite was run.
- The total number of scenarios and the success rate for the execution.
- For each scenario: the name of the scenario, status of the scenario, list of rules fired, list of executed ruleflow tasks, and the duration of the execution, if requested. If the scenario suite is a test suite, the list of tests with the status of each test and a message explaining the status. If the scenario suite is a simulation, the value of the KPI that was calculated during the simulation.

6.2.2 Runtime architecture

This section describes the following topics:

- Runtime components
- Runtime client API
- Scenario provider architecture
- KPI architecture

Runtime components

Figure 6-8 on page 131 shows the runtime components.

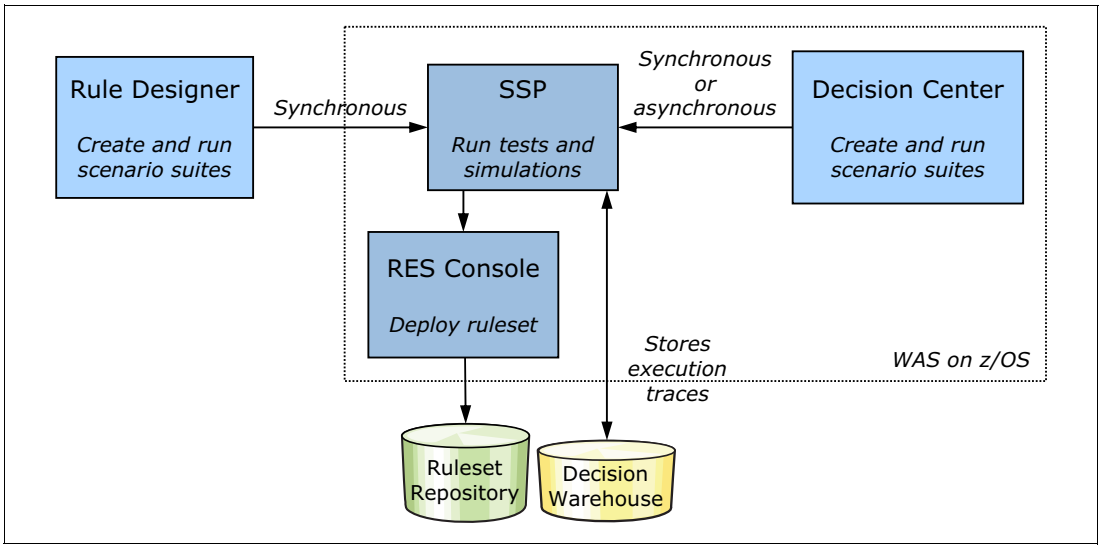


Figure 6-8 Testing and Simulation runtime components

At the heart of the runtime architecture for testing and simulation is the Scenario Service Provider (SSP) component. The SSP is a Java EE web application to be deployed into a WebSphere Application Server on z/OS application server (on the same server with Decision Center or another instance). The SSP provides a test and simulation runtime execution service to both Rule Designer and Decision Center. It can be deployed on multiple servers to dispatch the test and simulation load across several nodes.

The SSP is a stand-alone component. In heavy workloads, for example, when your business users plan to run many tests and simulations on a resource-intensive decision service, you can size the SSP independently of the Decision Center.

The SSP is a back-end component that does not provide a graphical user interface. A test or simulation execution is always triggered from either Rule Designer, Decision Center, or a Java application that uses the Java 2 Platform, Standard Edition (J2SE) testing and simulation runtime API.

The SSP provides both a synchronous and an asynchronous execution service. But, the asynchronous execution service is available from Decision Center and the testing and simulation runtime client API only (Figure 6-9).

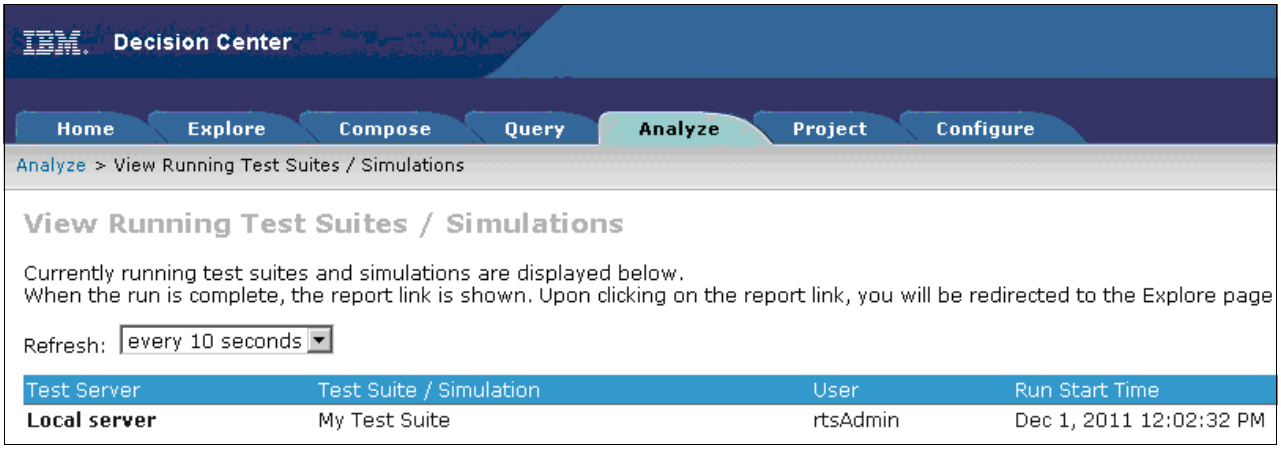


Figure 6-9 Monitoring asynchronous scenario suite execution in Decision Center

When a scenario suite execution is triggered, either synchronously or asynchronously, the SSP performs the following operations:

- Deploys the ruleset under test or simulation into the ruleset repository that is defined for the application server execution unit (XU)
- Executes each scenario case, using the input parameters that are defined in the scenario suite, using the XU component that is deployed in the application server
- If the scenario suite is a test suite, performs tests on the execution results
- If the scenario suite is a simulation suite, performs KPI calculations on the input parameters and execution results
- Undeploys the ruleset under test or simulation
- Returns an execution report

It is important to remember that to execute the ruleset under test or simulation, the SSP needs to have access to its Java Execution Module (XOM). Prior to running a scenario suite or a simulation on a ruleset, its XOM must be deployed in the XOM repository or repackaged into the SSP archive using the tooling provided in Rule Designer. You execute this operation only one time, unless the XOM is updated, in which case, it must be redeployed.

Because the SSP is not a native z/OS component, it benefits from the Decision Warehouse database that is available for executions using Decision Server in a WebSphere Application Server on z/OS application server. The SSP uses the Decision Warehouse database to store additional reporting data when requested by a business user in Decision Center. Whenever a business user requests the creation of an Excel file with the values of the output parameters, or any execution traces, such as the list of rules fired or the execution time, the corresponding traces are stored in the Decision Warehouse. The information stays in the Decision Warehouse so that the Decision Center database is not overloaded with extra report data.

Runtime client API

The Java runtime client API that is provided with WebSphere Operational Decision Management V7.5 allows a developer to write a Java application that runs an existing scenario suite either locally (in the same Java virtual machine (JVM), without deploying an SSP module) or against a deployed SSP. A typical use case is to write a batch that runs test suites and simulations every night and generates a complete execution report.

The main entry point in the runtime client API for running a scenario suite against a deployed SSP is the package ilog.rules.dvs.client. This package provides extensive online documentation with code samples that explain how to use it:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.dserver.reference.studio/html/api/html/ilog/rules/dvs/client/package-summary.html

For running a scenario suite locally, the class ilog.rules.dvs.ssp.IlrLocalSSPService is the main entry point. This class can be used with the previous code sample to replace the SSP service implementation returned by the SSP service factory.

Scenario provider architecture

The scenario provider is specified in the scenario suite format. The scenario provider is the low-level execution component that provides the scenario suite data in a suitable format for execution by the rule engine. The scenario provider provides the test component in the case of a test suite (Figure 6-10 on page 133).

Unless you want to use an existing data store (a relational database or a set of files, for example) to specify your scenario suite data, you do not write a custom scenario provider. You can reuse ilog.rules.dvs.core.scenarioproviders.IlrExcel2003ScenarioProvider, which is the Excel scenario provider, in all your scenario suite formats.

The code samples that are provided in this chapter demonstrate authoring a custom scenario provider that accesses a VSAM data store. In addition, WebSphere Operation Decision Center includes a more generic sample that explains how to retrieve scenario data from a relational database:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.dcenter.samples/Content/Business_Rules/_pubskel/Infocenter_Primary/ps_DC_Samples3185.html

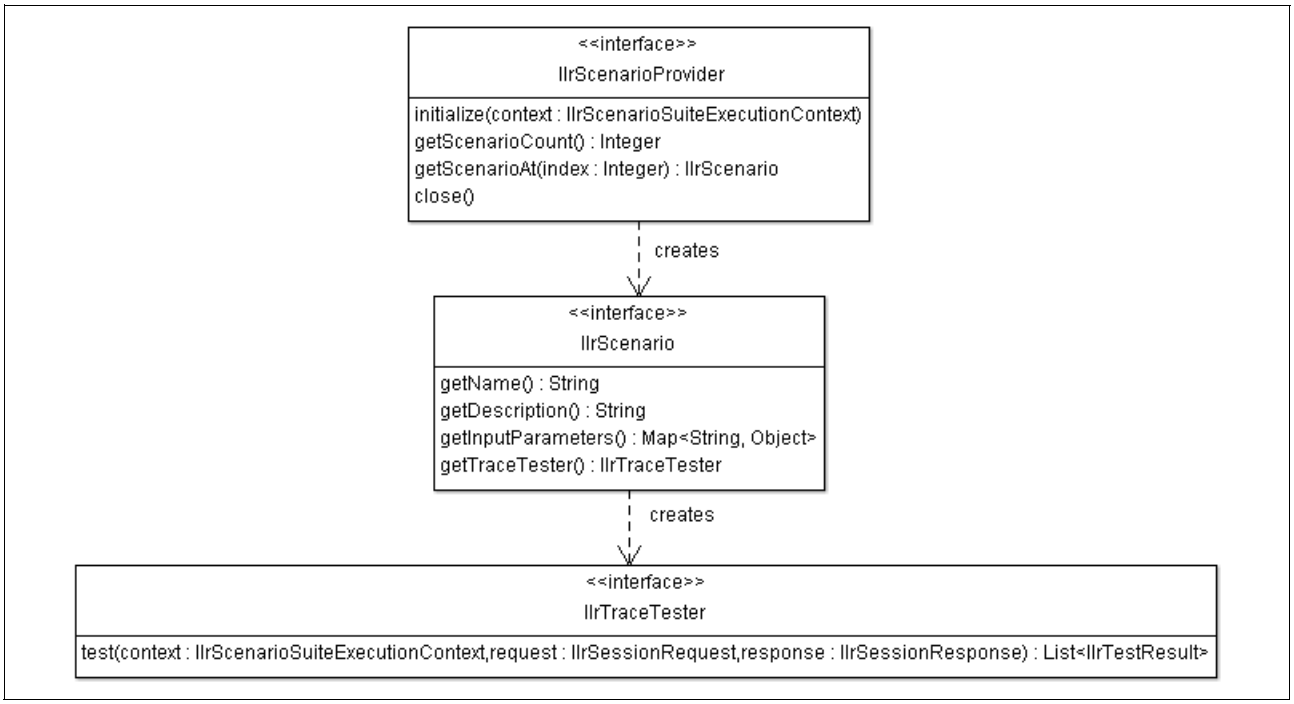


Figure 6-10 The scenario provider Java API

Scenario provider

A scenario provider is a Java class. It implements the ilog.rules.dvs.core.IlrScenarioProvider interface from the testing and simulation API. This component performs the following actions:

- Initialization: An initialization method that is provided by the component is called by the SSP before the execution of a scenario suite. This method can access a context that contains the data that the user entered in the Decision Center GUI (or in a Rule Designer launch configuration) when the user created the scenario suite. This way, the scenario that was provided can use this method to access the corresponding data store. For example, the Excel scenario provider that is provided with the Excel scenario suite formats uses this context to retrieve the Microsoft Excel file that was uploaded by the Decision Center user (or that was selected in the Rule Designer launch configuration) when the user created the scenario suite.
- Scenario count: After initialization, the component is expected to be able to return the total number of scenarios in the suite when requested by the SSP that is running it.
- Scenario retrieval: After initialization, the component is expected to be able to return any scenario from the suite. Each scenario is identified by its index in the suite.

- Resource cleanup: When requested, the component is expected to clean up any internal or external resource that it uses, such as a connection to a database.

The Decision Center public API defines a component that provides a GUI to a business user to initialize the scenario provider with data that is entered when a scenario suite is created, ilog.rules.teamserver.web.components.renderer.IlrScenarioSuiteResourcesRenderer. We present a sample implementation of this component later in this chapter.

Scenario

Every scenario in the suite is modeled by a scenario component, which is a Java class that implements the ilog.rules.dvs.core.IlrScenario interface from the testing and simulation API.

Primarily identified by its index in the scenario suite, a scenario also carries a name and description that can be useful for identifying its purpose within this scenario suite. Both the name and description are provided by the scenario component. Beside this identification data, the scenario component provides this important information:

- The map of Java input parameters to be used by the rule engine for the execution of the ruleset under test or simulation
- The trace tester component that verifies the execution result if the scenario is a test scenario

Trace tester

The trace tester component is a Java class that implements the ilog.rules.dvs.core.IlrTraceTester interface. When the SSP makes a request, the trace tester component performs tests on the execution result of a scenario to verify that this result conforms to the user expectation.

The trace tester component is created by the scenario component, which is created by the scenario provider component. The test specification is generally retrieved at the initialization of the scenario provider, as part of the scenario data.

KPI architecture

If authoring a custom scenario provider is not mandatory (unless you plan to use another data store than Microsoft Excel to define your scenario suites), authoring a custom KPI is mandatory if you want to run a simulation.

The test and simulation KPI API only defines two components, KPI and KPI result, that you can use for all your simulation needs (Figure 6-11 on page 135).

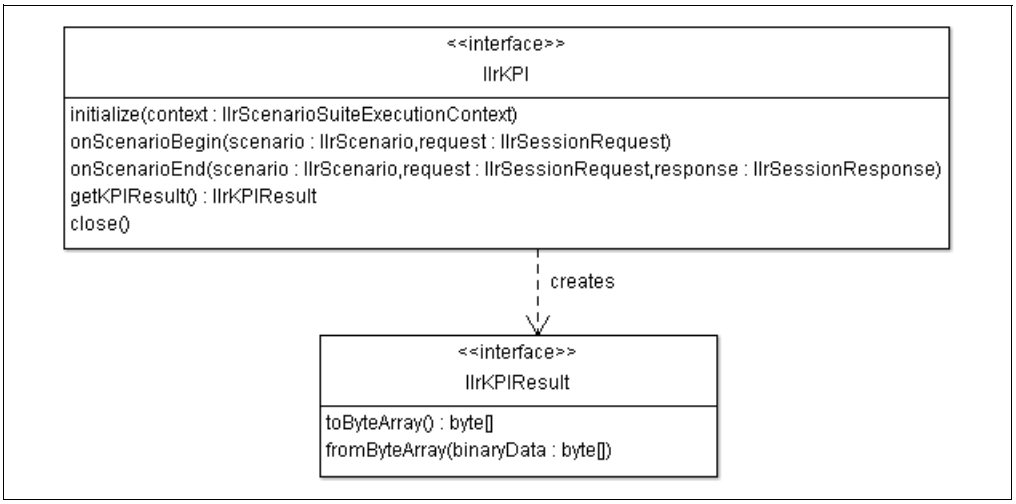


Figure 6-11 The test and simulation KPI Java API

Key performance indicator component

The KPI component is a Java class that implements the ilog.rules.dvs.core.IlrKPI interface. This component implements the following behavior:

- Initialization: An initialization method that is provided by the KPI component is called by the SSP before the execution of a scenario suite. This method has access to the same context as the scenario provider. This capability can be useful to retrieve information that is entered by the user when the user created the simulation to share a single KPI implementation/scenario suite format between multiple rule projects, using user data to decide how the KPI calculation needs to be performed for the current simulation.
- Beginning of a scenario: The KPI component is notified by the SSP each time that a scenario is going to be executed by the rule engine. To perform its calculation, if needed at this stage of the processing, the KPI component has access to the scenario that is going to be executed and to the technical request that is going to be sent to the rule engine.
- End of a scenario: The KPI component is notified by the SSP each time that a scenario is executed by the rule engine. To perform its calculation, the KPI component has access to the scenario that is executed and to the execution result (execution response) that is provided by the rule engine. Generally, at this stage, the KPI implementation performs its most useful calculations, based on the result of the scenario execution.
- End of the simulation: At the end of the simulation, when all scenarios are executed, the SSP requests that the KPI component return a KPI result for the simulation.

Key performance indicator result

The KPI result component encapsulates the value that is calculated by a KPI component for a simulation. This encapsulation makes it easy for the underlying system to carry the value through the modules of WebSphere Operational Decision Manager.

The KPI result component is a Java class that implements the interface ilog.rules.dvs.common.output.IlrKPIResult. It implements two symmetrical methods that are able to encode the KPI result as an array of bytes and to decode it.

Two common KPI result implementations are predefined for your use. The ilog.rules.dvs.common.output.impl.IlrKPIResultInteger encapsulates a KPI result that is an integer value. The ilog.rules.dvs.common.output.impl.IlrKPIResultString encapsulates a KPI result that is a string value.

To be able to display a KPI result in a simulation report, you must author one more component, ilog.rules.teamserver.web.components.renderers.IlrScenarioSuiteKPIRenderer. This component is not necessary if the simulations are run using only the testing and simulation client API.

6.2.3 Development and authoring tools

This section describes the development and authoring tools.

Rule Designer

You can perform the following operations that relate to testing and simulation from within Rule Designer by using the provided tooling:

- Check that a rule project is compatible with the Excel scenario suite formats. If a project is not currently compatible because of its BOM structure or a lack of verbalization, a list of actions to perform on the project to make it compatible is provided.
- Generate an Excel workbook to define a scenario suite for a rule project using one of the Excel scenario suite formats. The generated Excel workbook includes a help page that explains how to use it to define a scenario suite.
- Deploy the XOM (or repackage the SSP with the XOM) prior to the first execution of a scenario suite for a ruleset.
- Create a custom scenario suite format manually or by extending the existing scenario suite formats, using a set of specialized wizards and editors.
- Deploy a custom scenario suite into the SSP and Decision Center to make it available to the business users.
- Run a test suite using a launch configuration.

Decision Center

You can perform the following operations that relate to testing and simulation from within Decision Center by using the provided tooling:

- Enable or disable a custom scenario suite format for a rule project
- Create a test suite
- Create a simulation
- Run a test suite or simulation against all the rules of a rule project or against a subset of the rules
- Display a previous execution report for a test suite or simulation
- Compare two execution reports side by side

6.3 Testing and simulation lifecycle

Table 6-1 on page 137 lists the typical actions to perform during the lifecycle of a rule project for testing and simulation.

Table 6-1 Actions to perform during the lifecycle of a rule project

When	Who	What
During the early development of the decision project, before making the rule project available to the business user in the Decision Center.	Developer	► Create the rule project in Rule Designer. ► Check that the project is compatible with the scenario provider to run tests and a simulation; make the project compatible if necessary. ► Develop custom scenario suite formats with custom KPIs using the wizards and editors in Rule Designer. ► Repackage the SSP and the Decision Center with the XOM of the rule project and the custom scenario format.
When the project is ready to be deployed in Decision Center for the first time.	Administrator	► Deploy the SSP and Decision Center on WebSphere Application Server on z/OS. ► Configure the URLs of the SSP servers to perform testing and simulation in Decision Center.
When the project is ready to be deployed in Decision Center for the first time.	Developer	Publish the rule project in Decision Center.
When the project is deployed in Decision Center.	Administrator	Enable the custom scenario suite formats for the rule project in Decision Center.
► When the project is deployed in Decision Center along with the scenario suite formats to test it or run simulations. ► When a problem occurs during a test run and support is needed from a developer.	Business user	► Create and run scenario suites. ► Export DVS archive for developer.
When requested by a business user.	Developer	Debug DVS archive.

6.4 Running tests and simulations from Rule Designer and Decision Center

This section describes how to run tests and simulations from Rule Designer and Decision Center.

6.4.1 Scenario: Testing the insurance eligibility project using the Excel scenario suite format from Rule Designer and Decision Center

This scenario demonstrates the following functions on a use case:

- Create an Excel test suite in Rule Designer

- Run the Excel test suite in Rule Designer and display the execution report
- Repackage the SSP with the insurance eligibility project XOM and redeploy the SSP
- Publish the insurance eligibility project in Decision Center
- Configure Decision Center to use the SSP for running tests and simulations
- Create a test suite in Decision Center, run it, and display the execution report

The prerequisite to this scenario is to install zRule Execution Server for z/OS and Decision Center, including the Testing and Simulation modules, in a WebSphere Application Server instance on z/OS.

Step 1: Create an Excel test suite in Rule Designer

Perform the following steps to create an Excel test suite in Rule Designer:

1. In Rule Designer, start by opening the **insurance-rules** project that was created in Chapter 3, “Getting started with business rules” on page 27, as shown in Figure 6-12.

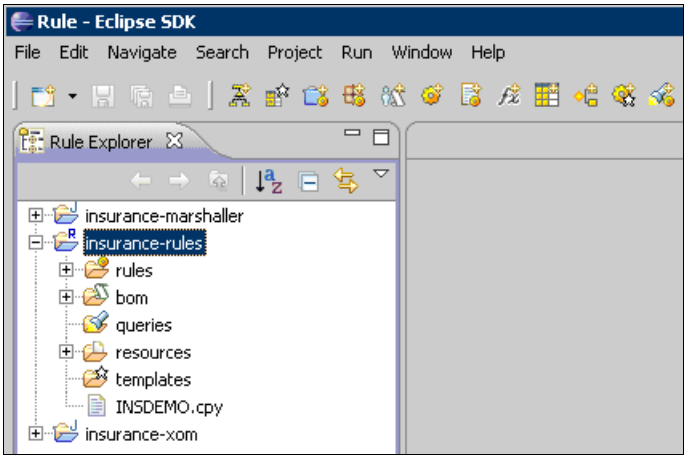


Figure 6-12 The insurance-rule project opened in the Rule Explorer

2. First, check that the BOM of this project is compatible with the Excel scenario suite format that we use to define a test suite for the project. To check, right-click the **insurance-rule** project and select **Decision Validation Services** → **Check Project**, as shown in Figure 6-13.

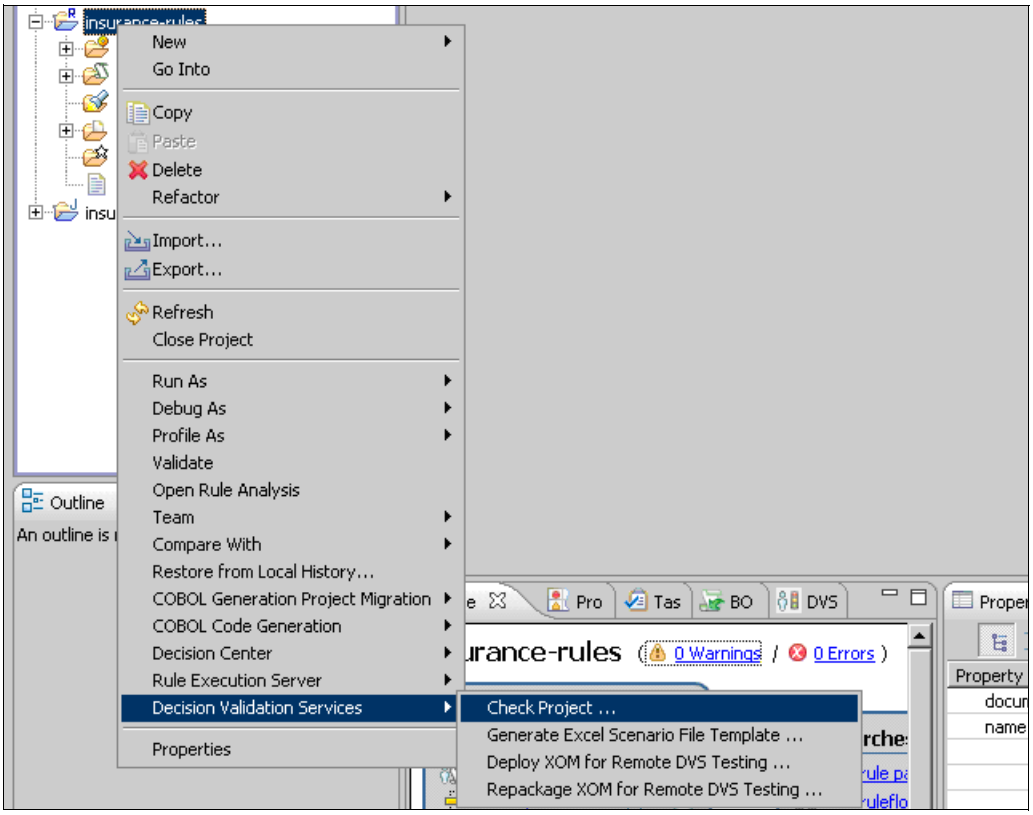


Figure 6-13 Check Project

A new view, DVS Project Validation, opens in Rule Designer. It displays a list of incompatibilities between the project and the Excel scenario suite formats that are provided for testing and simulation, along with the actions to resolve these incompatibilities. For the insurance-rules project, no incompatibilities are displayed in the DVS Project Validation view, as shown in Figure 6-14, which means that the project is compatible with all the Excel scenario suite formats.

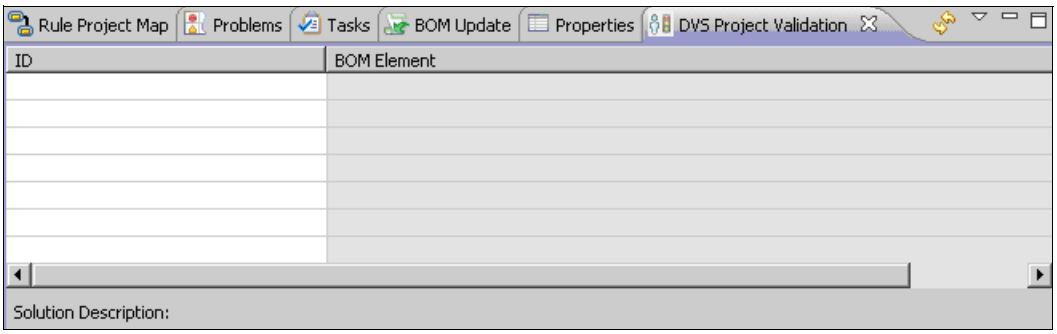


Figure 6-14 DVS Project Validation view that displays no incompatibilities

3. Now that we are sure that our project is compatible with the Excel scenario suite formats, we generate an Excel workbook to define our test suite. To generate the Excel workbook, right-click the **insurance-rules** project and select **Decision Validation Services** → **Generate Excel Scenario File Template**. This action opens a new wizard entitled Generate Excel Scenario File Template, which we use to specify the types of tests that we want to define in the Excel scenario suite.

   The first panel of the wizard displays the name of the selected rule project for which the scenario suite is created, as shown in Figure 6-15. Click **Next**.

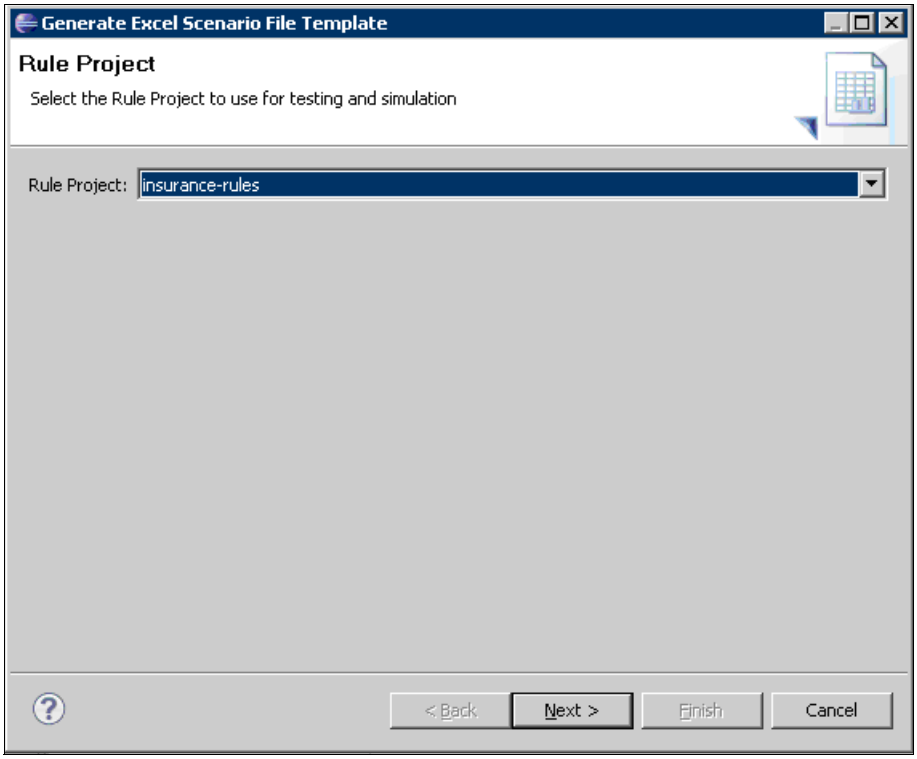


Figure 6-15 The first panel of the Generate Excel Scenario File Template wizard

4. The second panel, Generation Settings, displays the options for the Excel scenario suite format, as shown in Figure 6-16.

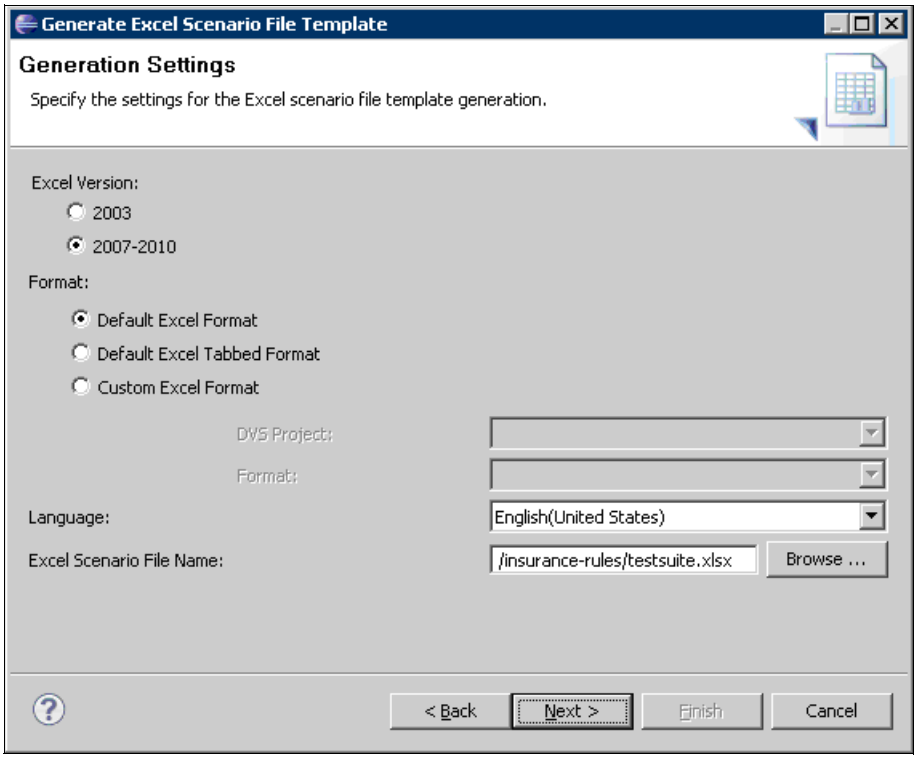


Figure 6-16 Excel Scenario File Template Generation Settings panel

In this window, select the version of Microsoft Excel to use to edit the test suite (2003 or **2007-2010**). Leave the **Default Excel Format** selected. Select the Language in which to generate the test suite (we use a language for which the insurance-rule project is verbalized): **English(United States)**. Then, click **Next**.

5. In the Expected Results panel, which is shown in Figure 6-17, we define the set of tests that we want to perform on the output of the insurance-rule ruleset.

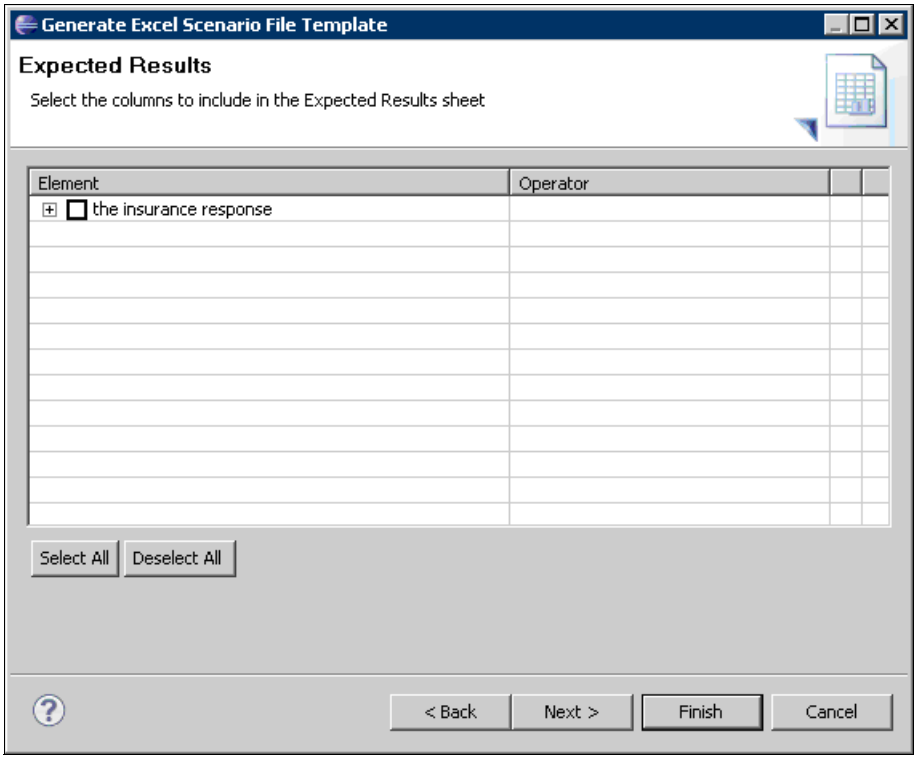


Figure 6-17 Expected results for an Excel test suite

6. For this scenario, select a test on the **approved** attribute of the insurance response, with the operator **equals**, and select a test on the **messages** attribute of the insurance response, with the operator **contains**. You select these tests by checking the corresponding attributes in the panel and, then, in the Operator column, by selecting the operator to use, as shown in Figure 6-18.

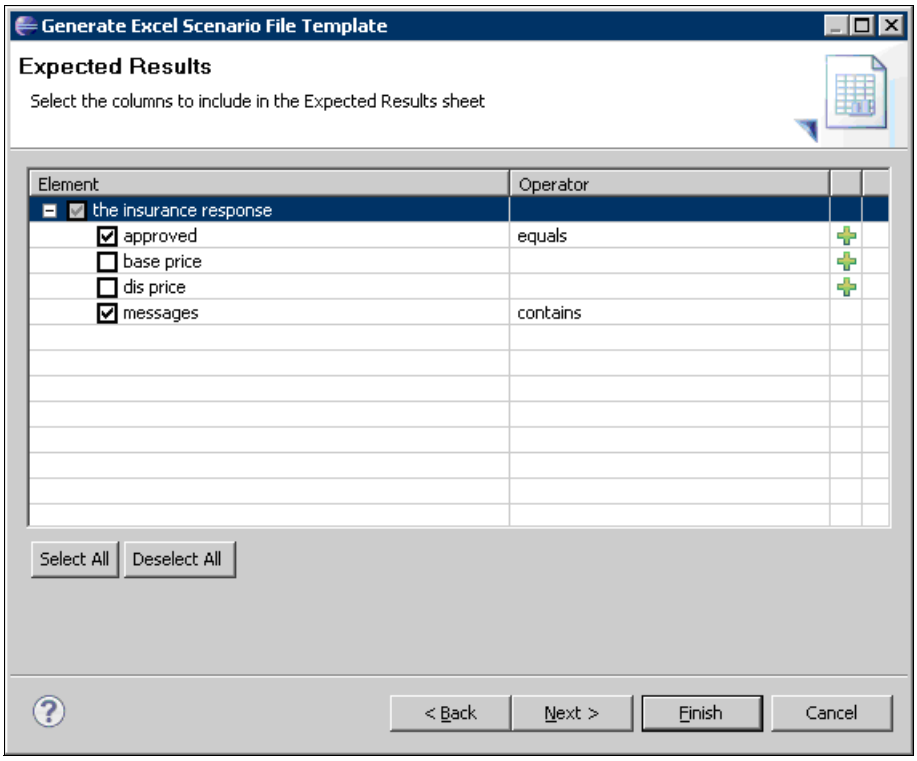


Figure 6-18 The Expected Results panel after the selection of our tests

7. To add tests on the Expected Execution Details panel, click **Next**, as shown in Figure 6-19.

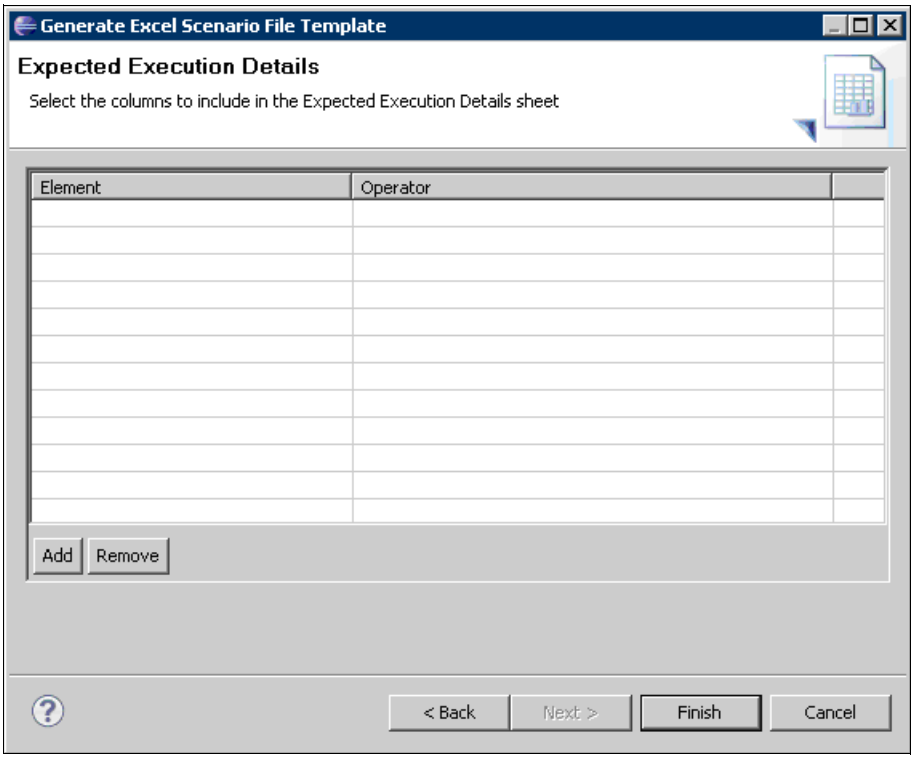


Figure 6-19 Expected Execution Details panel of the Generate Excel Scenario File Template wizard

8. To add a test on one execution detail, click **Add**, as shown in Figure 6-20.

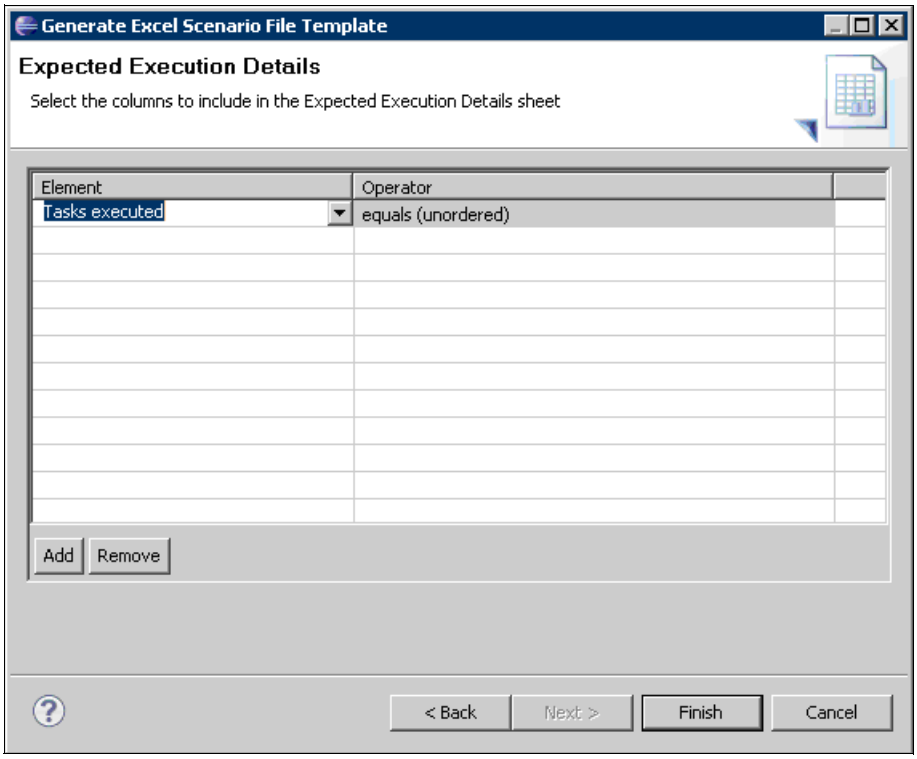


Figure 6-20 Adding a test on the execution details

9. We test which rules are fired for each scenario. In the Element column, select the **Rules fired** entry. Then, select the **contains** operator in the Operator column (Figure 6-21). Click **Finish**.

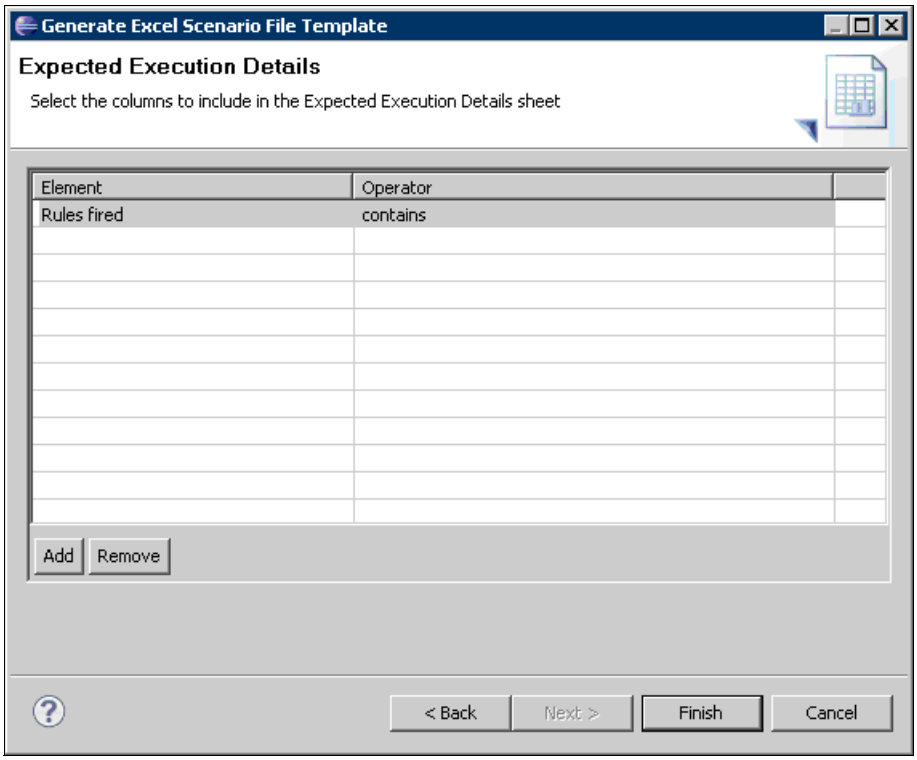


Figure 6-21 A "contains" test is defined for the list of rules fired

10. The file `testsuite.xls` is generated in the insurance-rules project directory, as confirmed by a message in the console view (Figure 6-22).

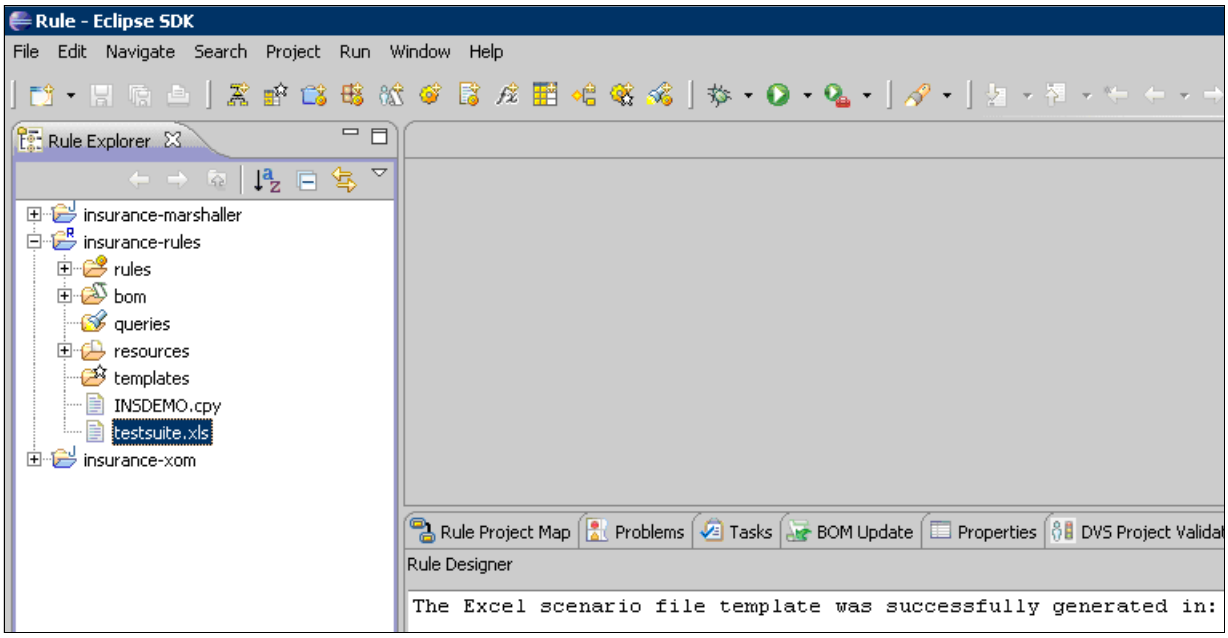


Figure 6-22 The testsuite.xls file generated in the insurance-rules project

11. To open the file for editing in Microsoft Excel, right-click **testsuite.xls** in the Rule Explorer and select **Open With** → **System Editor**.

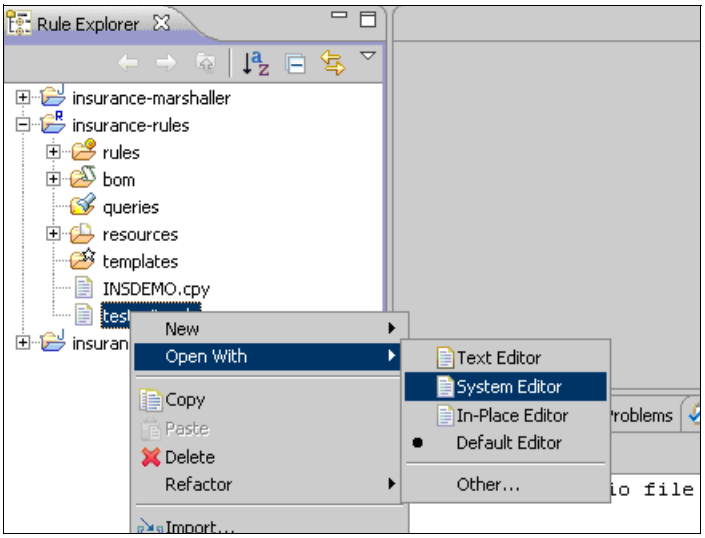


Figure 6-23 Selecting the Open With System Editor menu option

12. In Microsoft Excel, you can see that the generated workbook contains four worksheets, as shown by the tabs in Figure 6-24:
    - Scenarios worksheet to enter data for each of your test scenarios
    - Expected Results worksheet to optionally enter tests on the output parameters for each scenario
    - Expected Execution Details worksheet to optionally enter tests on the execution details for each scenario
    - HELP worksheet, which provides an embedded help page to help you define your test suite in the workbook

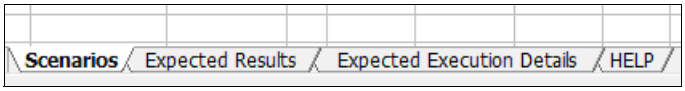


Figure 6-24 The worksheets in an Excel test suite

13. Start by using the Scenarios worksheet to create the first scenario for the insurance-rules ruleset. The Scenario worksheet displays a table in which each row represents a test scenario. For each row, the first column of the table is to define the name of the scenario, and the second column is to define an optional description for the scenario. The other columns are to define values for the input parameter attributes of the scenario. By default, the worksheet displays an empty row to be filled with the values for the first scenario (Figure 6-25).

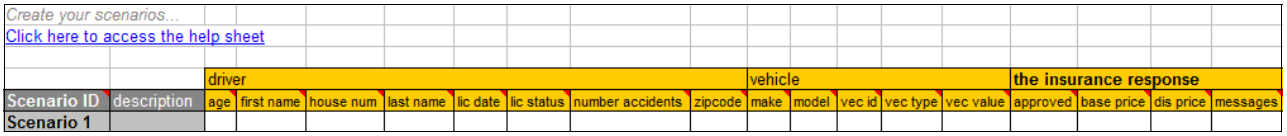


Figure 6-25 Initial content of the Scenarios worksheet

14. Enter the values in the Scenario worksheet, as shown in Figure 6-26, to define two similar scenarios that differ by the age of the driver. Both scenarios (Scenario 1 and Scenario 2) are supposed to trigger the execution of the rule validation.MaxiMinimumAge.

Create your scenarios...
Click here to access the help sheet

		driver								vehicle					the insurance response			
Scenario ID	description	age	first name	house num	last name	lic date	lic status	number accidents	zipcode	make	model	vec id	vec type	vec value	approved	base price	dis price	messages
Scenario 1	less than 18	17	John		Doe	09/08/2011	F	0	XA123456	BMW	GOLF	ABC12345678	SU	2000000				
Scenario 2	more than 60	61	Steve		Smith	01/12/1974	F	0	XA123457	BMW	GOLF	ABC12345679	SU	200000				

Figure 6-26 Two test scenarios for the insurance-rules ruleset

15. The next step is to specify the tests on the output parameter for each scenario. Display the Expected Results worksheet by clicking the **Expected Results** worksheet tab at the bottom of the Excel workbook. Enter the rows that are shown in Figure 6-27 to define our expectations for Scenario 1 and Scenario 2:
   - For Scenario 1, type `FALSE` and `The age exceeds the maximum or minimum`.
   - For Scenario 2, type `FALSE` and `The age exceeds the maximum or minimum`.

Fill only the cells for the results you want to test...
Click here to access the help sheet

Scenario ID	the insurance response is approved equals	the messages of the insurance response equals (unordered)
Scenario 1	FALSE	The age exceeds the maximum or minimum
Scenario 2	FALSE	The age exceeds the maximum or minimum

Figure 6-27 Testing the output parameter of the insurance-rules ruleset

16. The last step is to specify the tests on the execution details. Display the Expected Execution Details worksheet by clicking the **Expected Execution Details** worksheet name at the bottom of the Excel workbook. Enter the rows that are shown in Figure 6-28:
   - For Scenario 1, type `validation.MaxiMinimumAge`.
   - For Scenario 2, type `validation.MaxiMinimumAge`.

Fill only the cells for the execution details you want to test...
Click here to access the help sheet

Scenario ID	the list of fired rules contains
Scenario 1	validation.MaxiMinimumAge
Scenario 2	validation.MaxiMinimumAge

Figure 6-28 Testing the list of rules fired for the insurance-rules ruleset

17. Now, save the content of the Excel workbook and proceed to the next step, which is running the test suite in Rule Designer and displaying the execution report.

Step 2: Run the Excel test suite in Rule Designer and display the execution report

Perform the following steps to run the Excel test suite in Rule Designer and display the execution report:

1. To run an Excel test suite in Rule Designer, we must create a run configuration, which is also called a *launch configuration*. To create a run configuration, click **Run** → **Run Configurations**, as shown in Figure 6-29 on page 148.

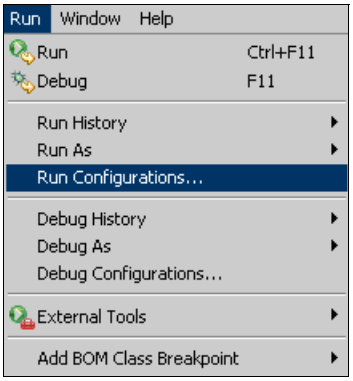


Figure 6-29 The Run Configurations menu

2. The Run Configurations Editor opens, as shown in Figure 6-30.

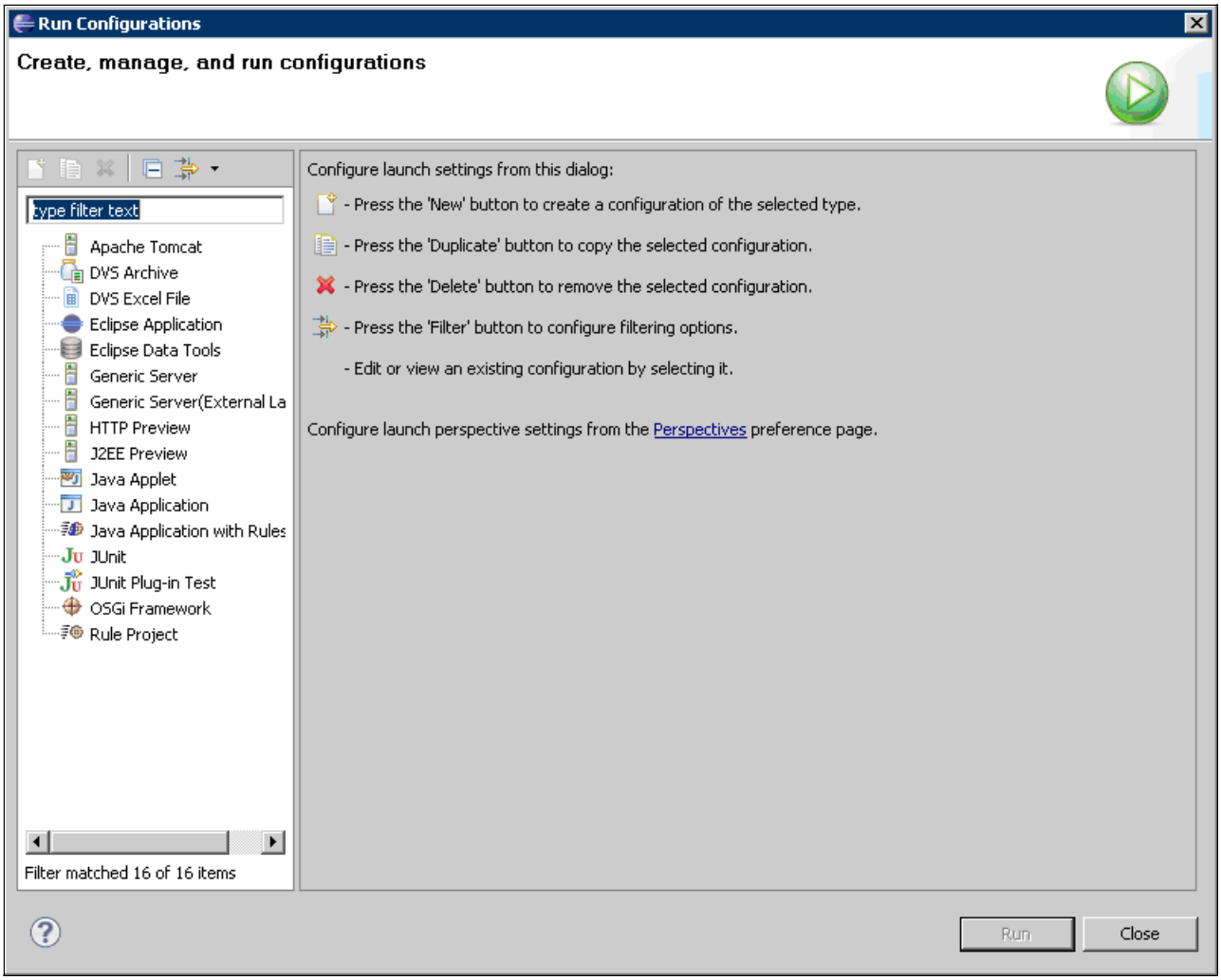


Figure 6-30 The Run Configurations Editor

3. To create a run configuration for the Excel test suite in this editor, select the **DVS Excel File** entry in the left section of the editor. Then, click the **New Launch Configuration** icon (the leftmost icon in the editor toolbar), as shown in Figure 6-31.

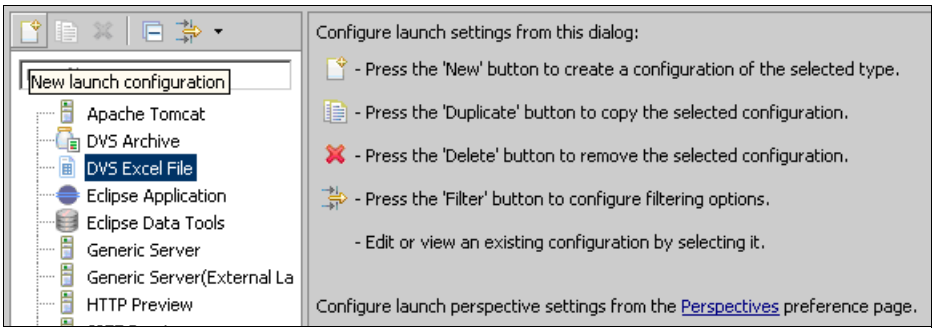


Figure 6-31 Selecting the DVS Excel File type of Run Configurations

4. The right side of the editor is updated to display a dedicated editor for DVS Excel file run configurations (Figure 6-32).

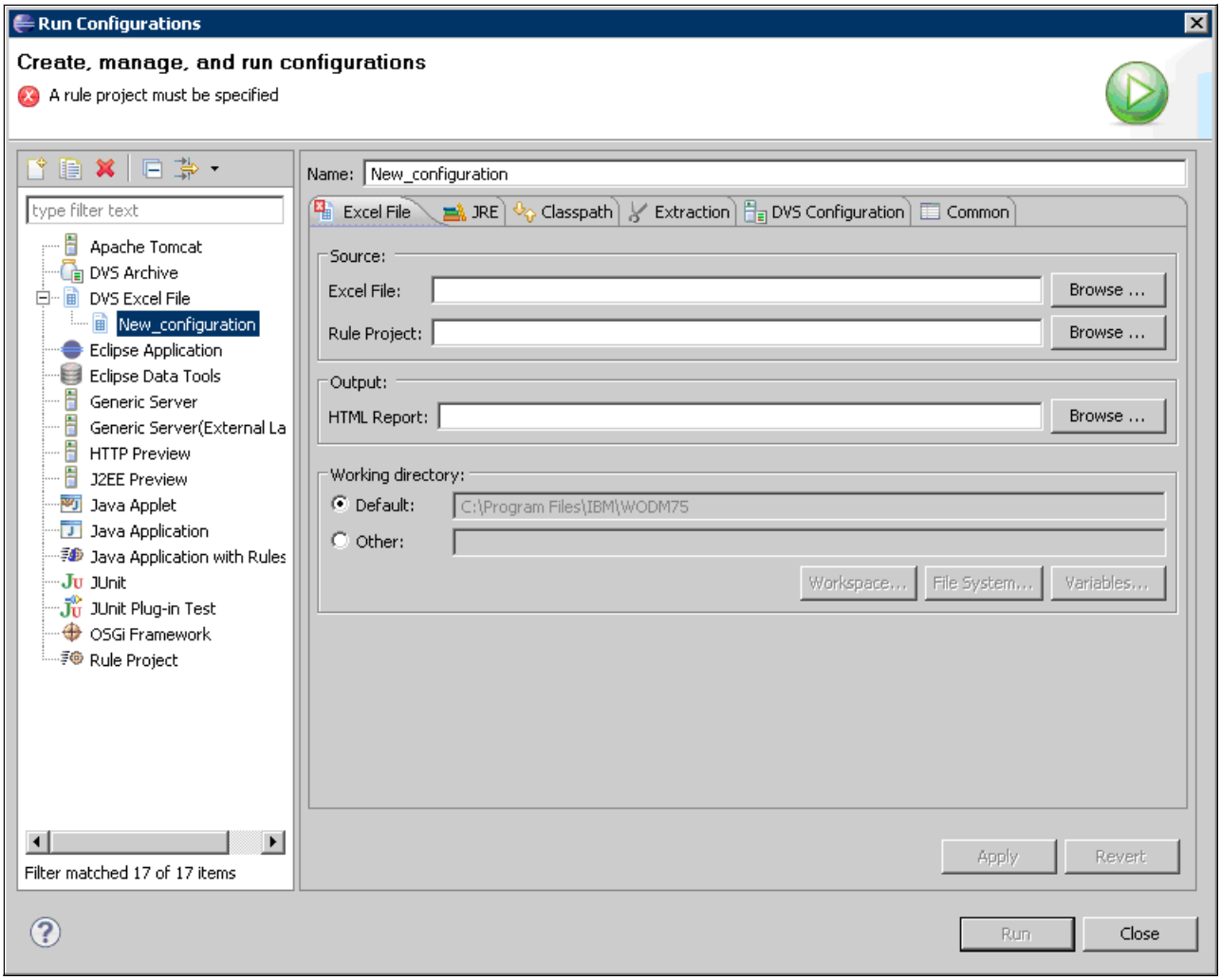


Figure 6-32 DVS Excel File Run Configurations Editor

5. Update the name of this new run configuration from New_configuration to `Run test suite against insurance rules`. Click **Browse** to select the following information for this run configuration (Figure 6-33):
   - For Excel File, select **\insurance-rules\testsuite.xls**.
   - For Rule Project, select **insurance-rules**.
   - For HTML Report, select **\insurance-rules\report.html**.

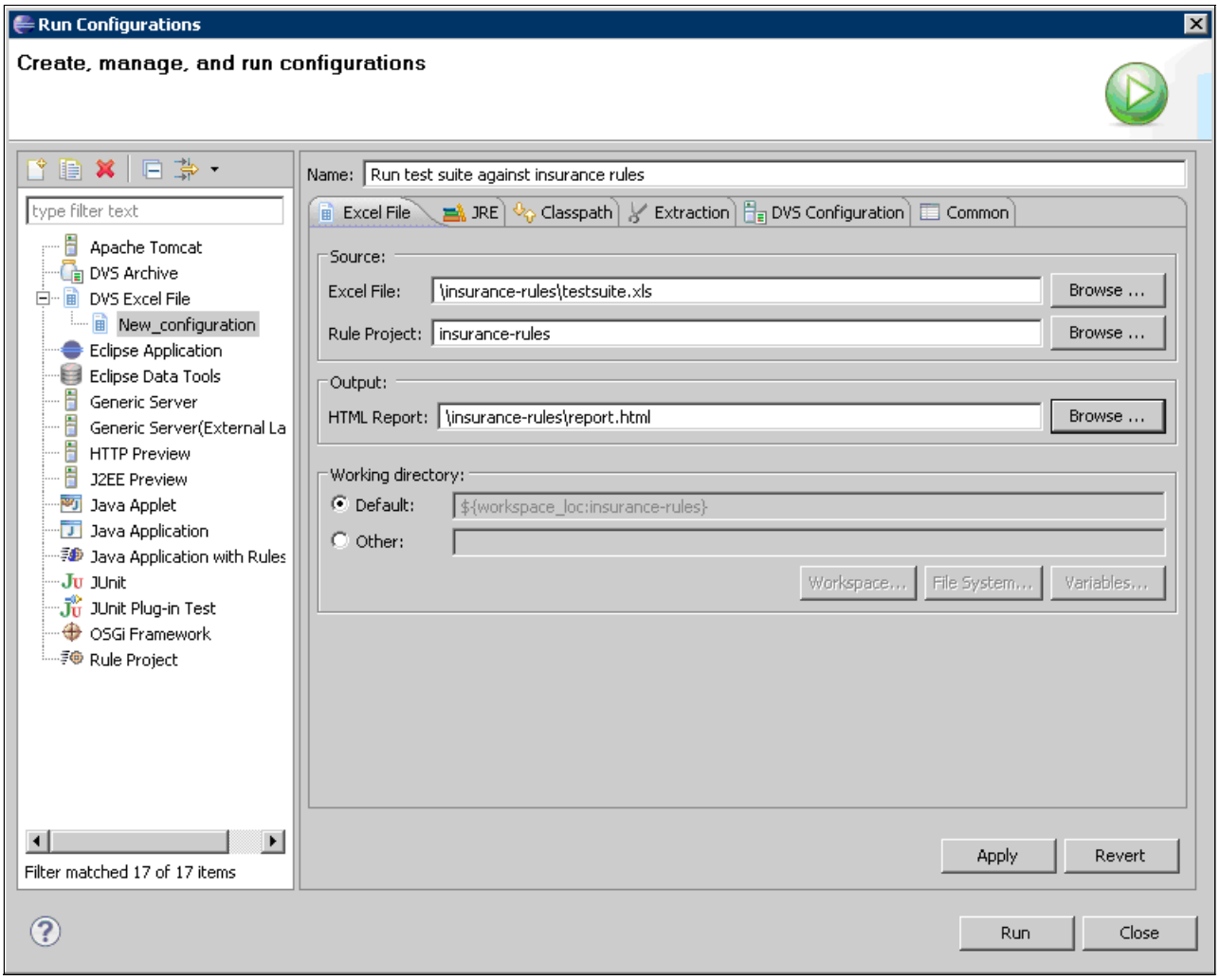


Figure 6-33 The run configuration for the insurance-rules Excel test suite

Click **Apply** to save the run configuration. Then, click **Run** to execute the test suite. The execution starts.

6. The content of the Console view is updated in real time with the execution logs for the test suite (Figure 6-34).

```
Rule Project Map | Problems | Tasks | BOM Update | Properties | DVS Project Validation | Console | Synchronize
<terminated> Run test suite against insurance rules [DVS Excel File] C:\Program Files\IBM\WODM75\jdk\bin\javaw.exe (Dec 15, 2011 10:28:46 AM)
Dec 15, 2011 10:28:49 AM ilog.rules.res.xu.log.internal.IlrLogHandler write
INFO: Logging started Decision Server - 7.5.0.0 - 2011-10-22 15:40:10
Dec 15, 2011 10:28:49 AM ilog.rules.res.xu.log.internal.IlrLogHandler write
INFO: JDK Logging level is set to: INFO.
Dec 15, 2011 10:28:49 AM ilog.rules.res.persistence.impl.file.IlrFileRepositoryDAO setConfigParameters
INFO: XOM repository set in file persistence mode: C:\Users\Administrator\IBM\WODM\DS\redbookws\insurance-rules\res_xom
--- Output for scenario 'Scenario 1' :

--- Output for scenario 'Scenario 2' :

Execution finished
```

Figure 6-34 Test suite execution logs in the Console view of Rule Designer

7. When the message "`Execution finished`" is displayed in the Console view log (Figure 6-34), the HTML execution report is generated. Display the HTML execution report to see the result of the test suite execution. To see the report in the Rule Explorer, first refresh the content of the insurance-rules project so that Rule Designer checks for new files in the project directory. To refresh the content, right-click the **insurance-rules** project in the Rule Explorer and select **Refresh** (Figure 6-35).

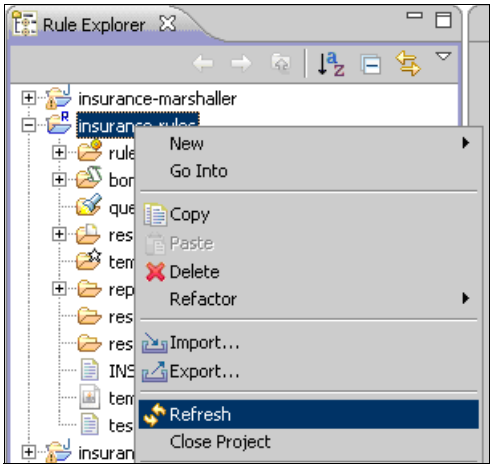


Figure 6-35 Refresh menu option for the insurance-rules project

8. After the content of the rule project is refreshed, the report file `report.html`, which was generated after the execution of the test suite, is displayed in the Rule Explorer. Right-click **report.html** and select **Open With** → **Web Browser** to display the report in Rule Designer (Figure 6-36).

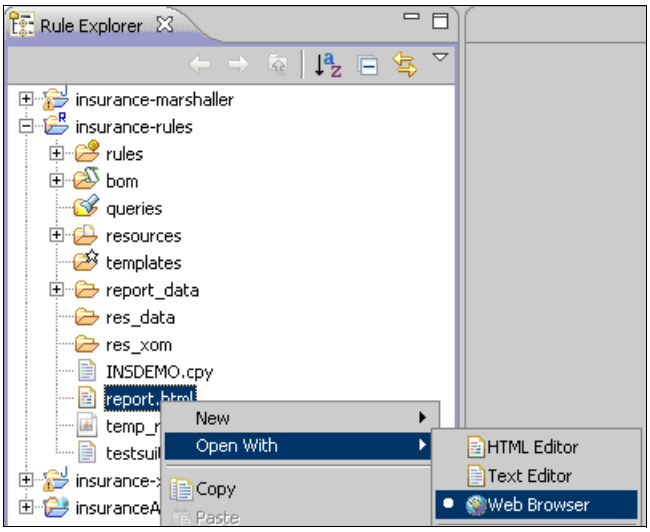
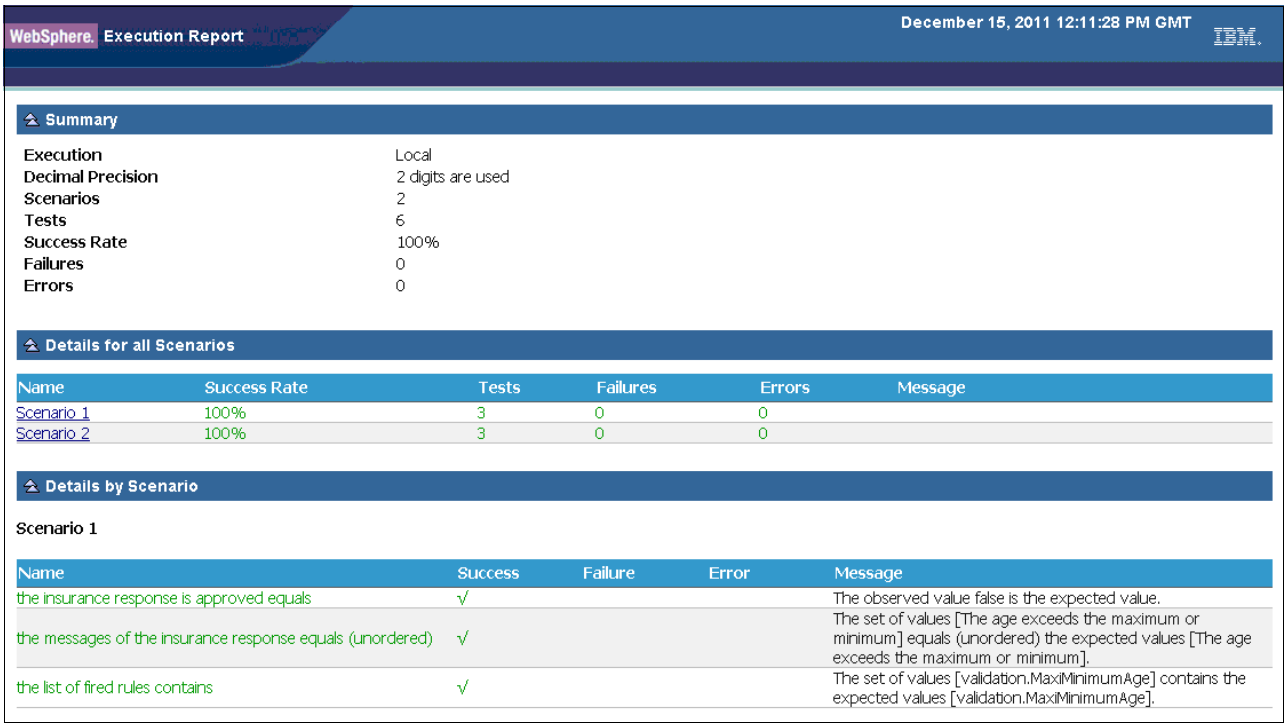


Figure 6-36 Open a test report with the Rule Designer Web Browser

9. The Execution Report opens in the internal web browser of Rule Designer. We see that the test suite executed successfully and that the test success rates are 100%, as shown in Figure 6-37.



WebSphere. Execution Report — December 15, 2011 12:11:28 PM GMT — IBM.

Summary

Execution	Local
Decimal Precision	2 digits are used
Scenarios	2
Tests	6
Success Rate	100%
Failures	0
Errors	0

Details for all Scenarios

Name	Success Rate	Tests	Failures	Errors	Message
Scenario 1	100%	3	0	0	
Scenario 2	100%	3	0	0	

Details by Scenario

Scenario 1

Name	Success	Failure	Error	Message
the insurance response is approved equals	√			The observed value false is the expected value.
the messages of the insurance response equals (unordered)	√			The set of values [The age exceeds the maximum or minimum] equals (unordered) the expected values [The age exceeds the maximum or minimum].
the list of fired rules contains	√			The set of values [validation.MaxiMinimumAge] contains the expected values [validation.MaxiMinimumAge].

Figure 6-37 Rule Designer Execution Report for the insurance-rules test suite

Step 3: Repackage the SSP with the insurance eligibility project XOM and redeploy the SSP

Follow these steps to repackage the SSP with the insurance eligibility project XOM and redeploy the SSP:

1. To execute the same test suite from Decision Center against the SSP, first repackage the XOM of the insurance eligibility project into the SSP and redeploy it. Create a new type of project, which is a DVS project that defines a customization for testing and simulation in the SSP and Decision Center. To create this project, right-click in the Rule Explorer white space and select **New** → **Other** (Figure 6-38).

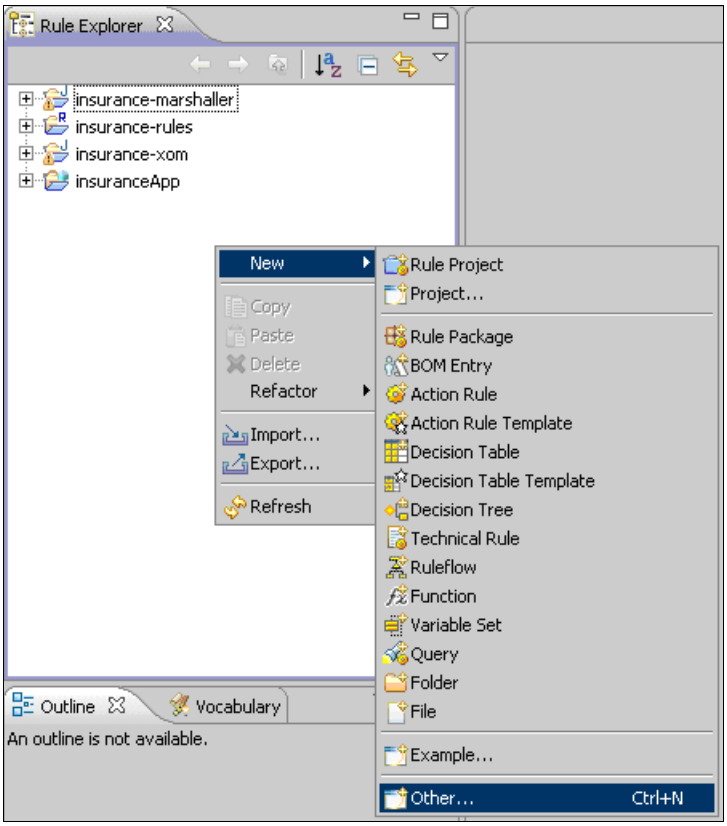


Figure 6-38 Selecting New → Other

2. This action displays the New wizard panel that prompts you to select a wizard. Select the wizard type **Rule Designer** → **Decision Validation Services** → **DVS Project** (Figure 6-39). Click **Next >** to display the first panel of the New DVS Project wizard.

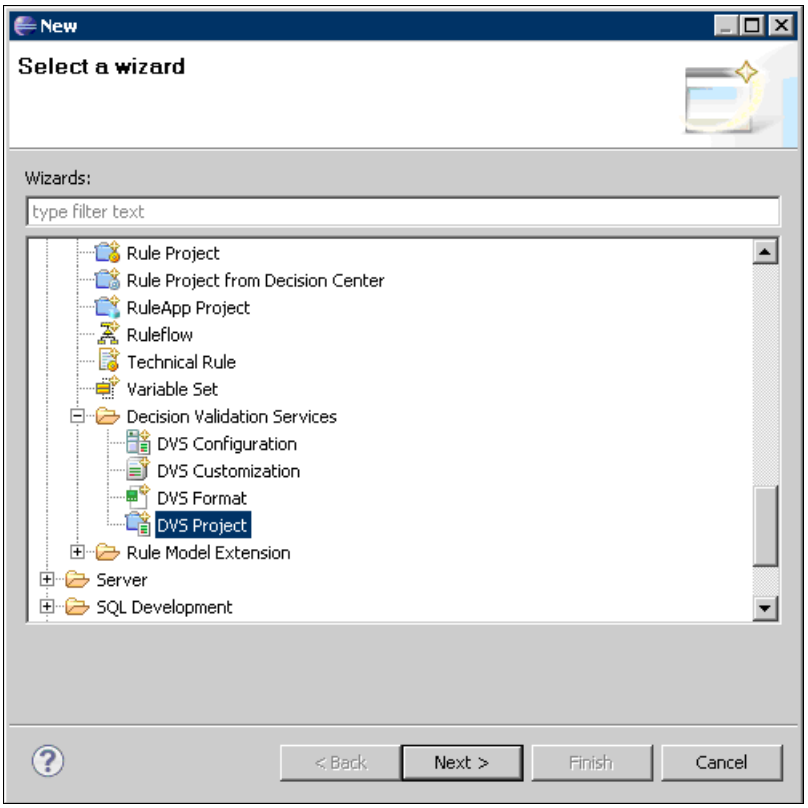


Figure 6-39 Selecting the new DVS Project wizard

3. This DVS Project panel prompts us to select a project name and location for the new project (Figure 6-40). Keep the default options and click **Next >**.

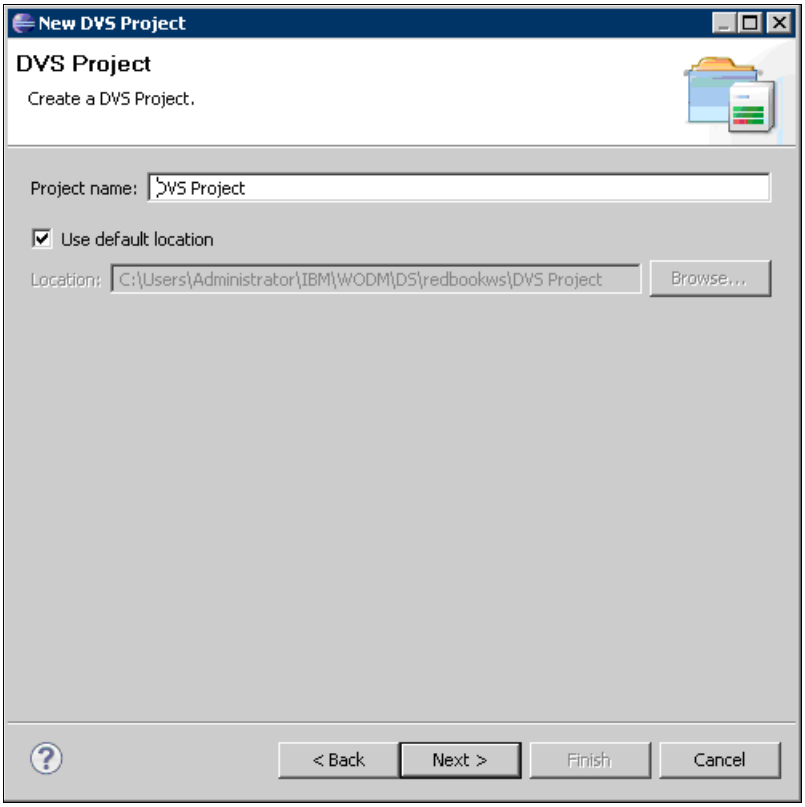


Figure 6-40 The first panel of the New DVS Project wizard

4. This panel prompts us for the name of the new customization to create in the project (Figure 6-41). Keep the default name for the customization and click **Finish**.

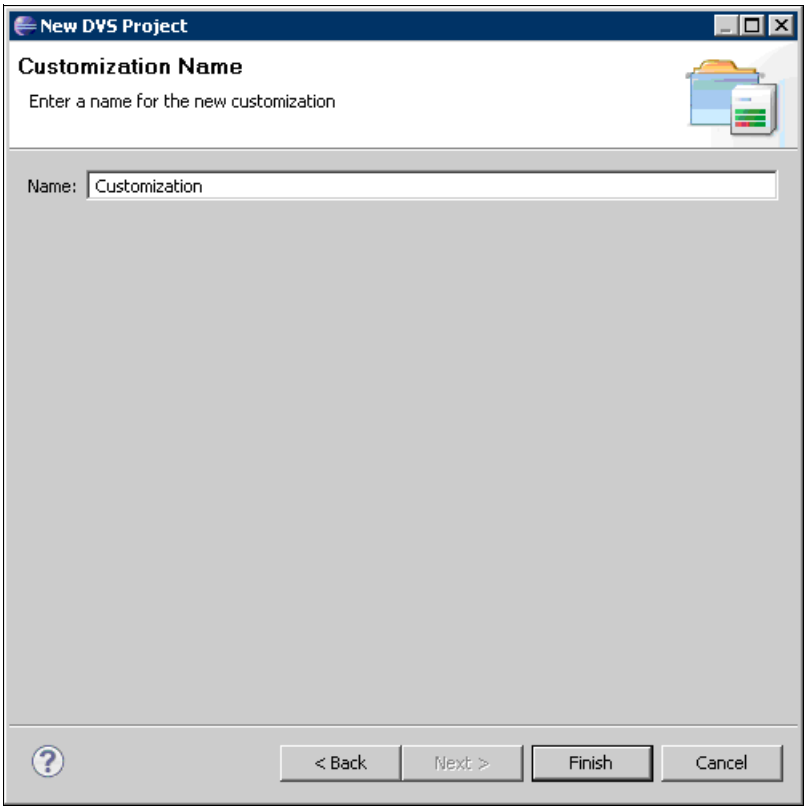


Figure 6-41 The second panel of the New DVS Project wizard

5. The DVS Customization Editor is then displayed in Rule Designer, allowing us to edit our new customization (Figure 6-42). First, add a server configuration to our customization so that the repackaging mechanism knows that we use WebSphere Application Server to host the SSP and Decision Center. On the DVS Customization page that is shown in Figure 6-42, in the Configurations section, click **Create**.

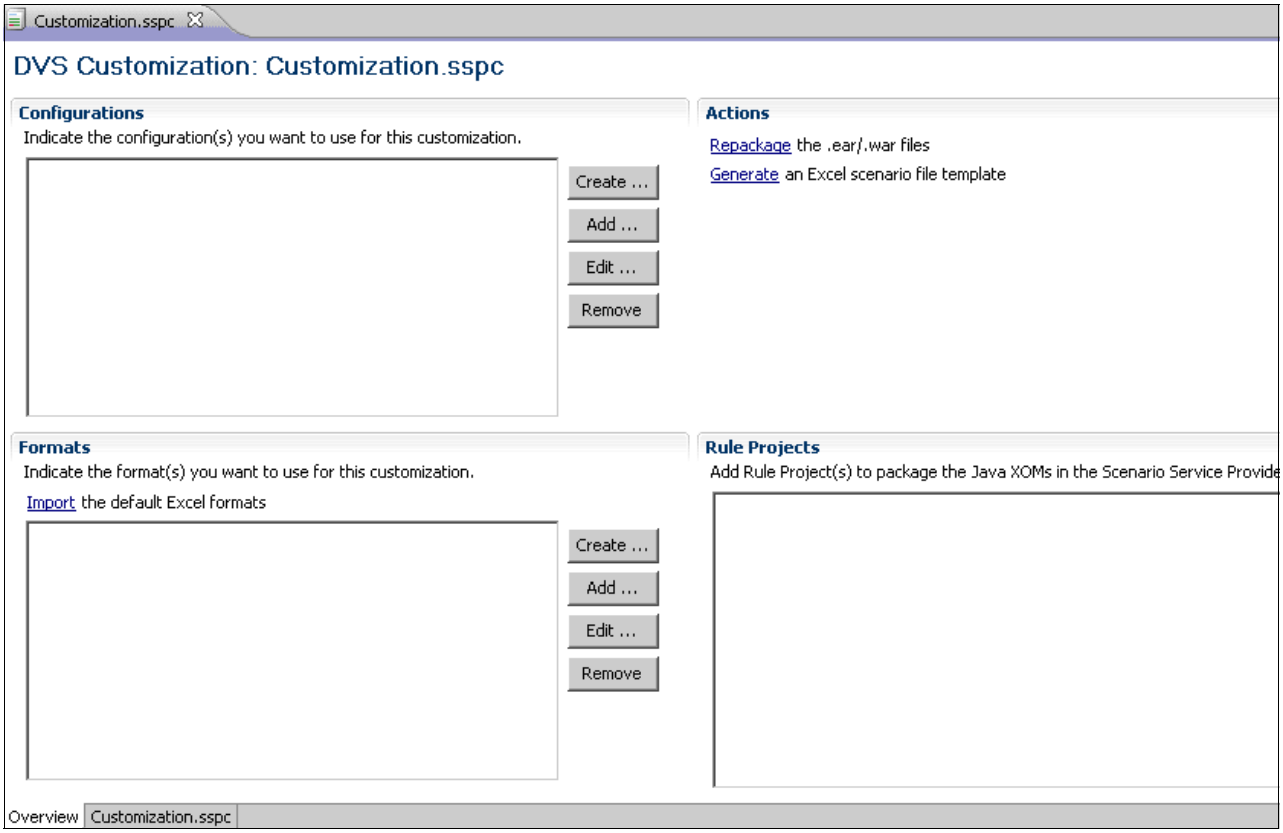


Figure 6-42 DVS Customization Editor

6. This action displays the first panel of the DVS Configuration wizard (Figure 6-43). On the Configure environment page that is shown in Figure 6-43, select **IBM WebSphere AS 7.0** as the application server on which to deploy the SSP and Decision Center, and click **Next**.

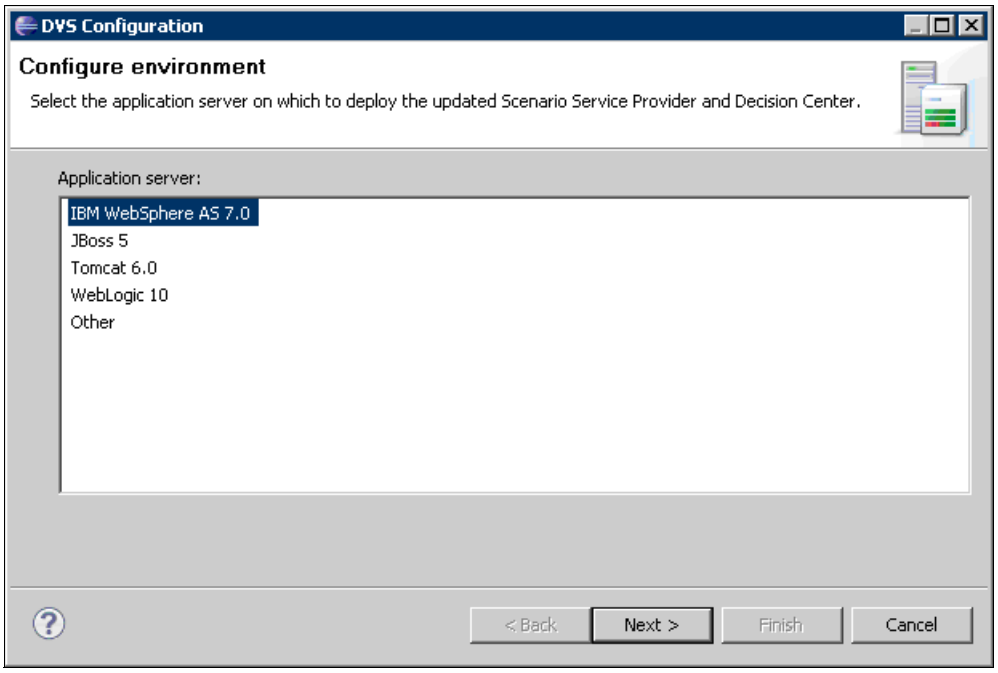


Figure 6-43 The first panel of the DVS Configuration wizard

7. The Configuration URLs panel opens. Enter the URL to our Scenario Service Provider instance, which is the URL to our WebSphere Application Server 7 server, ending with `/testing`). Enter the login and password credentials to connect to the SSP (Figure 6-44).

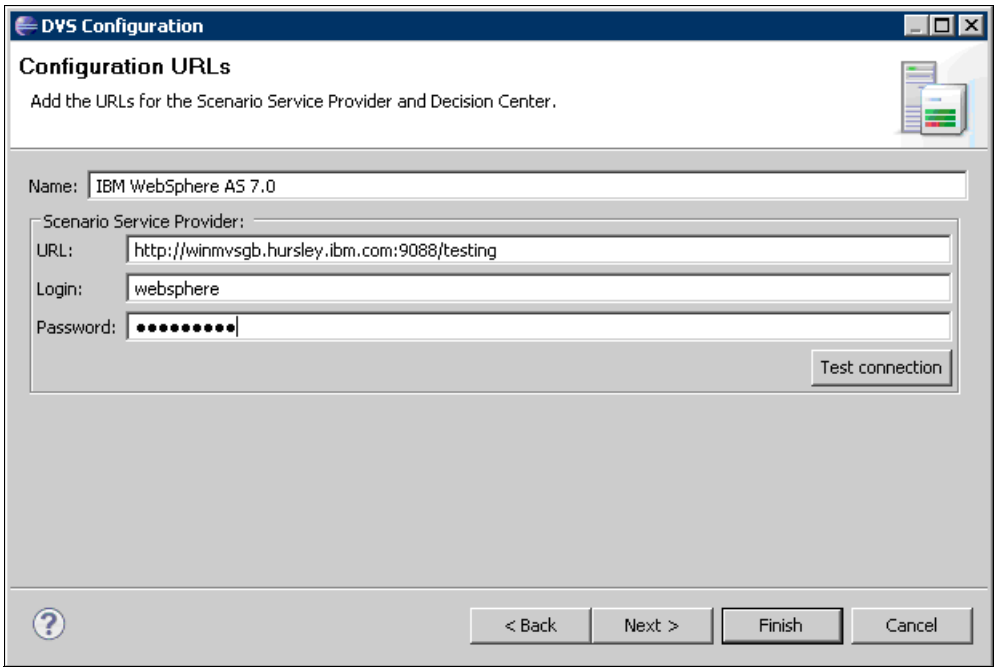


Figure 6-44 Configuration URL for a DVS configuration

On the Configuration URLs page that is shown in Figure 6-44 on page 158, we click **Test connection** to check that the SSP can be contacted with the provided URL and credentials. Then, we click **Finish**.

The DVS Configuration wizard is closed.

8. The DVS Configuration Editor opens for the WebSphere Application Server instance that we specified (Figure 6-45). On the DVS Configuration page that is shown in Figure 6-45, click the **Customization.sspc** tab to display the DVS Customization Editor, and press Ctrl+S to save its content.

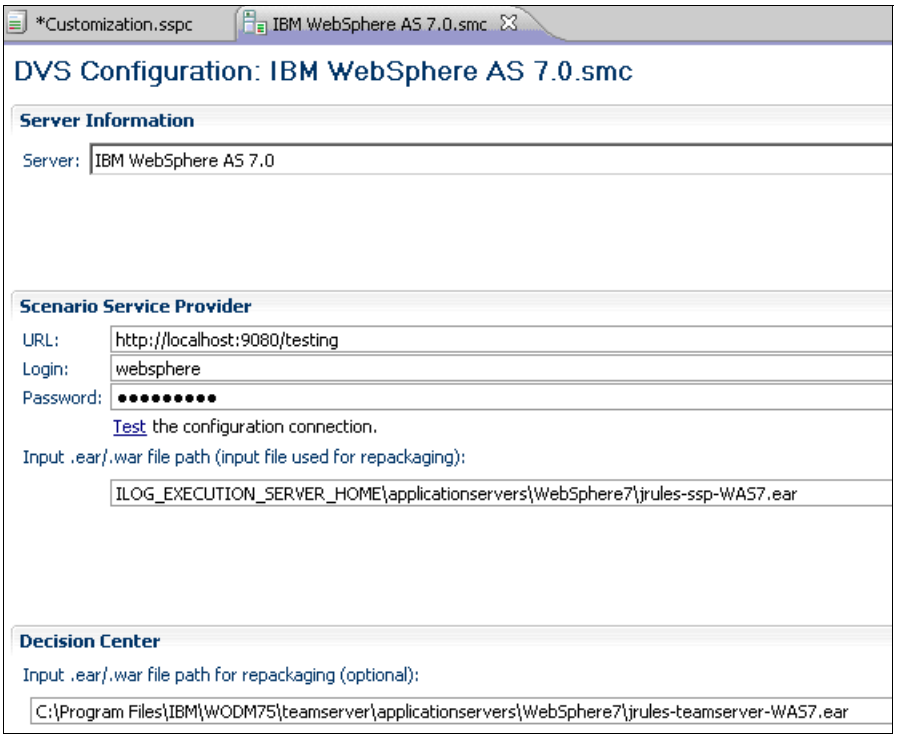


Figure 6-45 DVS Configuration Editor for a WebSphere Application Server 7.0 instance

9. The Configurations section is updated with the new server configuration that we defined (Figure 6-46 on page 160). We add the insurance-rules rule project to the customization so that Rule Designer is aware of the XOM to deploy to the SSP when repackaging it. Click **Add** in Configurations section of the editor.

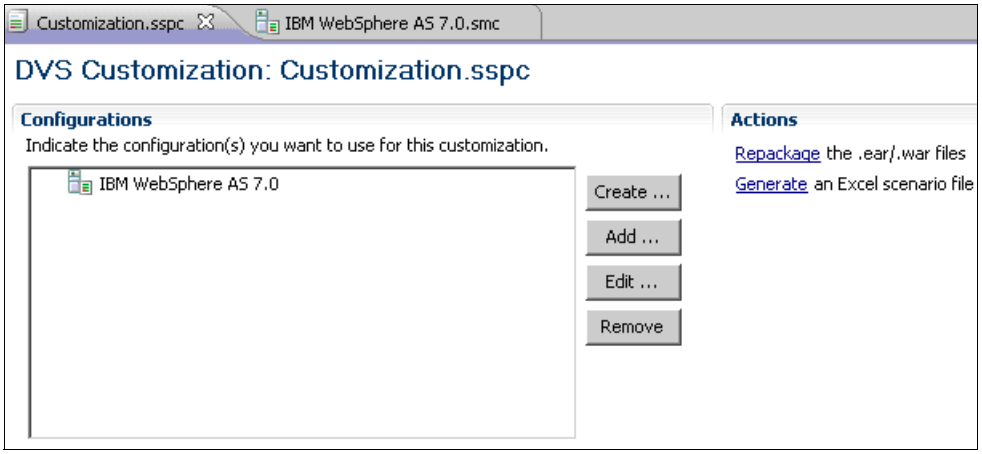


Figure 6-46 DVS customization updated with a new server configuration

10. A pop-up window opens for us to select the rule project to add to the customization (Figure 6-47). Select the **insurance-rules** project and click **OK** to add insurance-rules to the Rule Projects section of the Customization Editor.

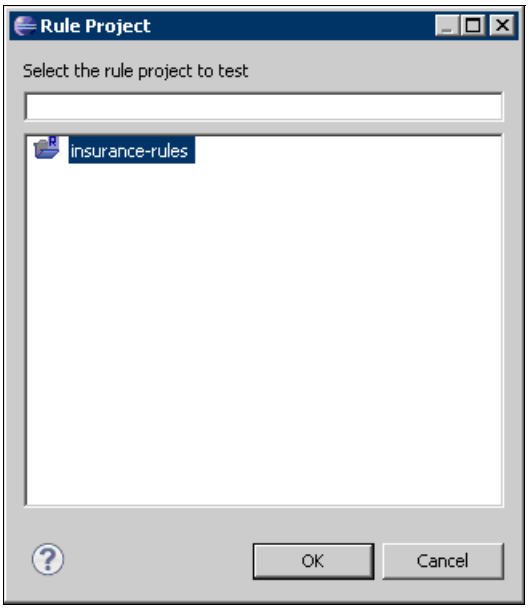


Figure 6-47 Selecting the rule project for the DVS customization

11. Figure 6-48 shows the insurance-rules project added to the Rule Projects section of the DVS Customization Editor. Save the content of the editor by pressing Ctrl+S. On the page that is shown in Figure 6-48, in the Actions section of the editor, click the **Repackage** the .ear/.war files link.

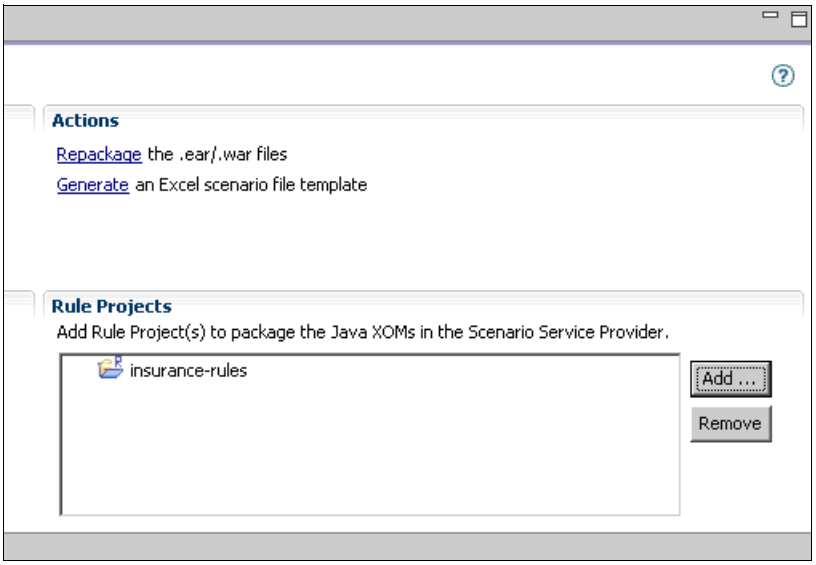


Figure 6-48 Insurance-rules project added to the Rule Projects section of the Customization Editor

12. A pop-up window opens so that we can specify the file to repackage (Figure 6-49). Because we repackage the SSP with the XOM of the insurance-rules project only (this option was already selected), clear the **Repackage Decision Center .ear/.war file** option, and click **OK**.

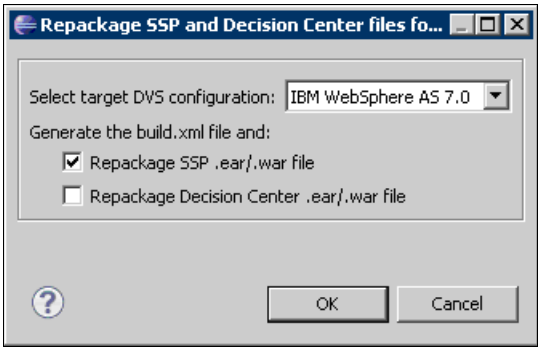


Figure 6-49 Option to repackage only the SSP

13. When the repackaging is complete, a message window opens to confirm the status of the repackaging operation. The last file path that is displayed in this message window informs us of the location of the repackaged SSP, which we redeploy in the WebSphere Application Server instance (Figure 6-50).

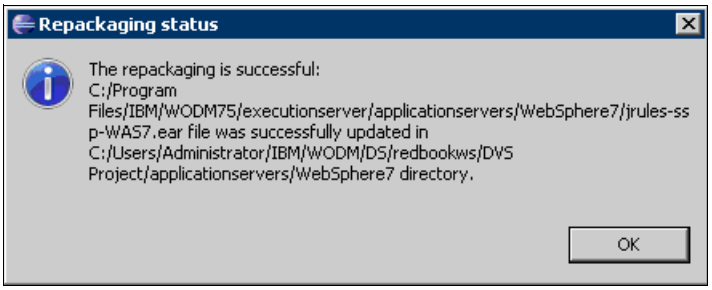


Figure 6-50 The message displayed after a successful repackaging of the SSP

Next, we redeploy the repackaged SSP, following the instructions in Configuring Decision Center for z/OS:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/index.jsp?topic=%2Fcom.ibm.dcenter.z.configuring%2FContent%2FBusiness_Rules%2F_pubskel%2FInfocenter_Primary%2Fps_DCz_Configuring3324.html

Step 4: Publish the insurance eligibility project in Decision Center

Follow these steps to publish the insurance eligibility project in Decision Center:

1. When the repackaged SSP is deployed in your application server, the next step is to publish the insurance-rules project in Decision Center so that the business users can access it and define a test suite for it. In the Rule Explorer, right-click the **insurance-rules** project and select **Decision Center** → **Connect** (Figure 6-51 on page 163).

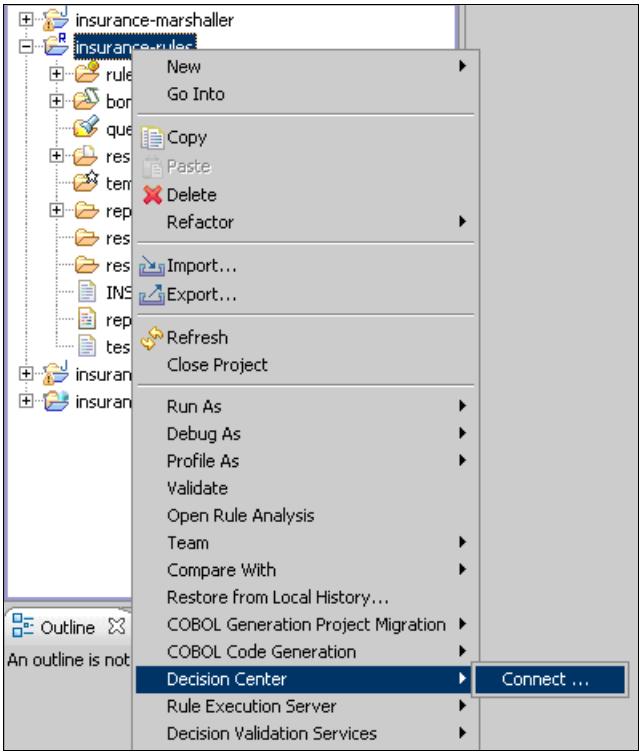


Figure 6-51 Selecting Decision Center → Connect

2. This action displays the Decision Center configuration windows. We must update this content so that the connection information matches the current Decision Center deployment (Figure 6-52). Enter the URL, user name, and password, and click **Connect**.

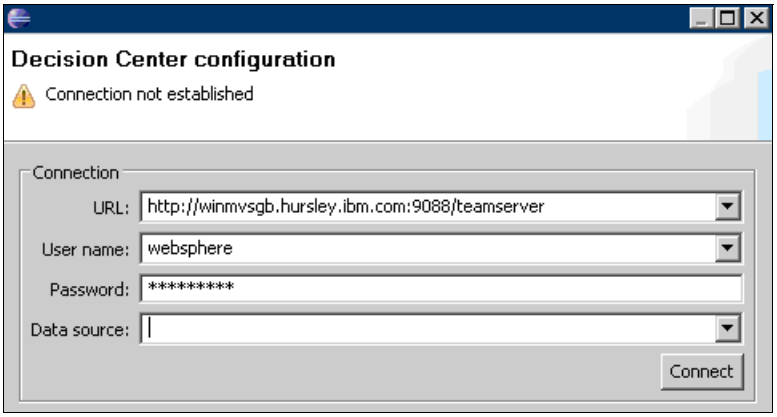


Figure 6-52 The connection information for a Decision Center configuration

3. Figure 6-53 shows that we established an actual connection. After the connection to Decision Center is successfully established, select the operation to perform in the Project configuration section (Figure 6-53). In this case, we want to create the project in Decision Center, so select **Create a new project on Decision Center** and click **Finish**.

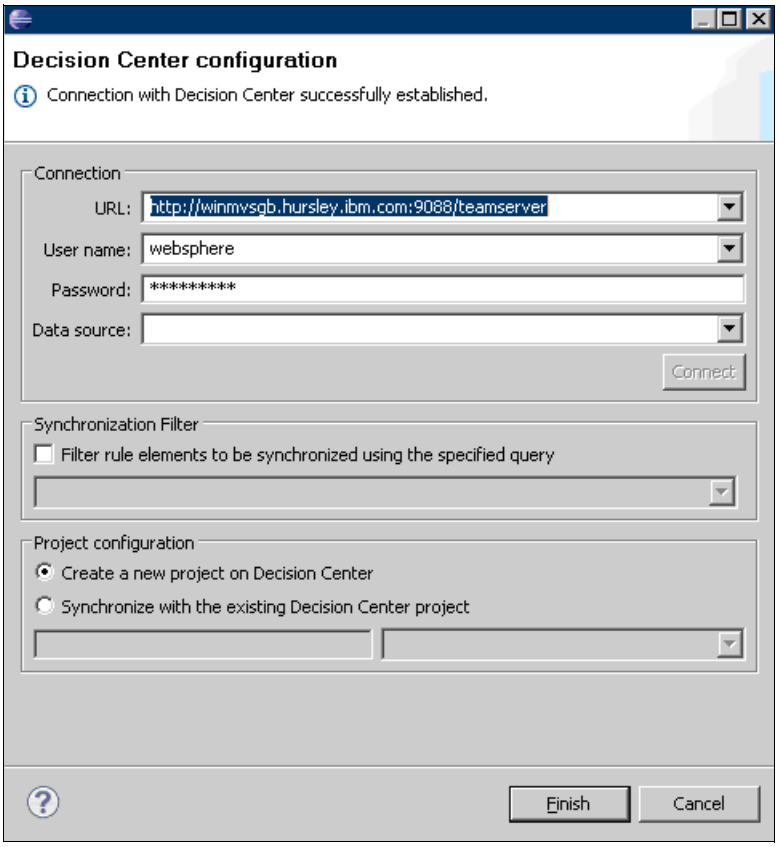


Figure 6-53 Creating a project on Decision Center from Rule Designer

4. A pop-up window opens to inform us of the progress of the publication of our project in Decision Center (Figure 6-54).

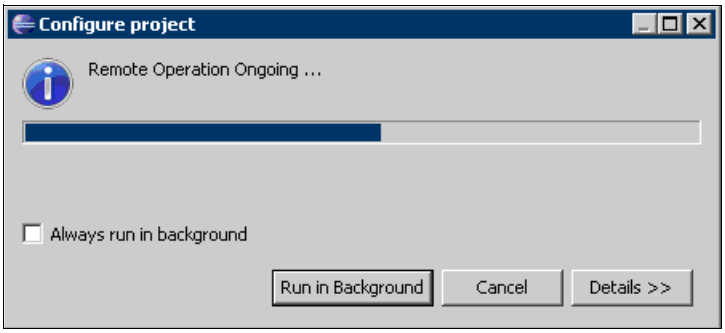


Figure 6-54 Publishing a project in Decision Center

5. After the publication to Decision Center is complete, the first pop-up window opens to give us the option to switch to the Team Synchronizing perspective (ignore this option and stay on the Rules perspective).

6. A second pop-up window opens to inform us that the synchronizing is complete. The projects are now the same in both Rule Designer and Decision Center (Figure 6-55 on page 165).

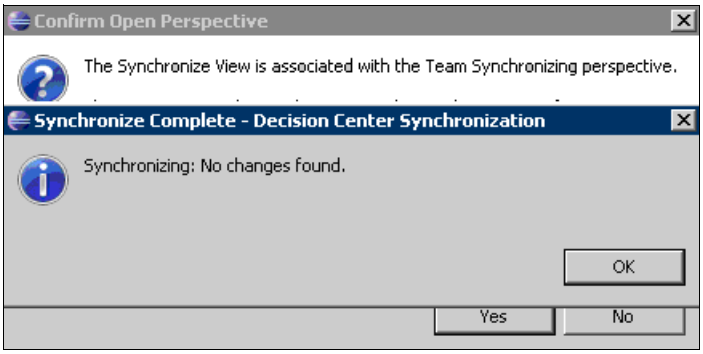


Figure 6-55 Confirmation that the publication in Decision Center is complete

Step 5: Configure Decision Center to use the SSP for running tests and simulations

Follow these steps to configure Decision Center to use the SSP for running tests and simulations:

1. Before the business users can define and run a test suite for the insurance-rules project in Decision Center, we must perform a few administrative operations. Open the browser to the Decision Center URL and sign in as a Decision Center administrator (Figure 6-56).

Figure 6-56 The sign-in page for Decision Center

2. On the Decision Center Home page, select the newly published **insurance-rules** project as the project in use (Figure 6-57 on page 166).

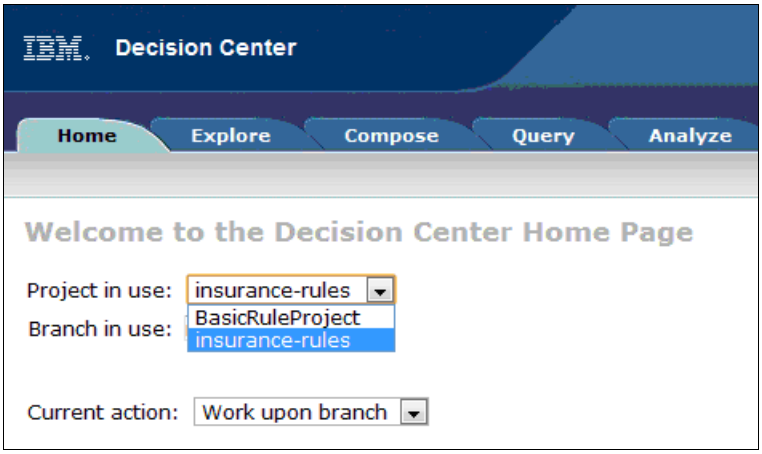


Figure 6-57 Selecting the insurance-rules project in Decision Center

3. When the insurance-rules project is selected as the current project, click the **Configure** tab to display the Configure menu for Decision Center (Figure 6-58).

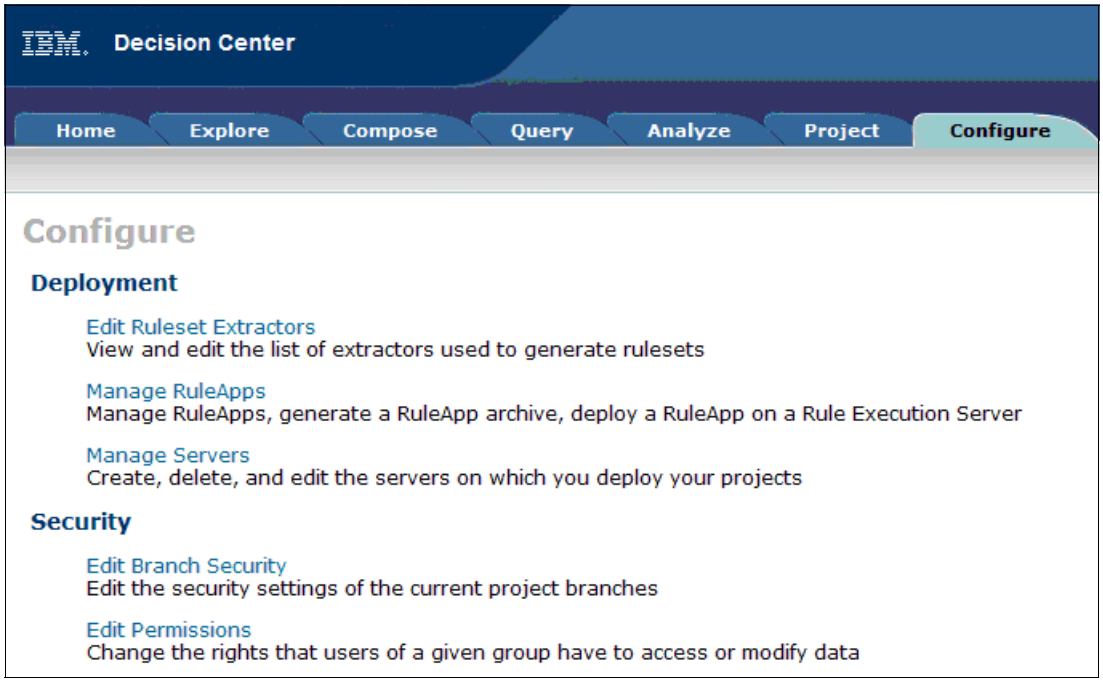


Figure 6-58 The Configure menu for Decision Center

4. First, we add the SSP to the list of servers known to Decision Center to run tests and simulation. *We perform this operation only one time, and then, the added SSP is available for all projects in Decision Center*. On Figure 6-58, click the **Manage Servers** link to display the Manage Servers page (Figure 6-59 on page 167).

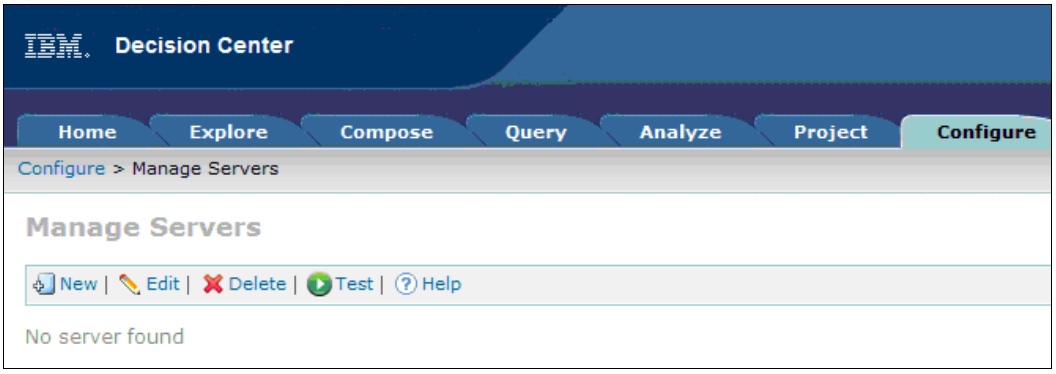


Figure 6-59 The Manage Servers page in Decision Center

5. In the Manage Servers toolbar (Figure 6-59), click **New** to create a server definition.
6. On the Create Server page that opens (Figure 6-60), enter the connection information for our server (URL and credentials). Select the following Usage options:
   - **Rule Execution Server**
   - **Both**

   Select the **All groups** Authorized groups option. When finished, click **OK** to save this new server definition.

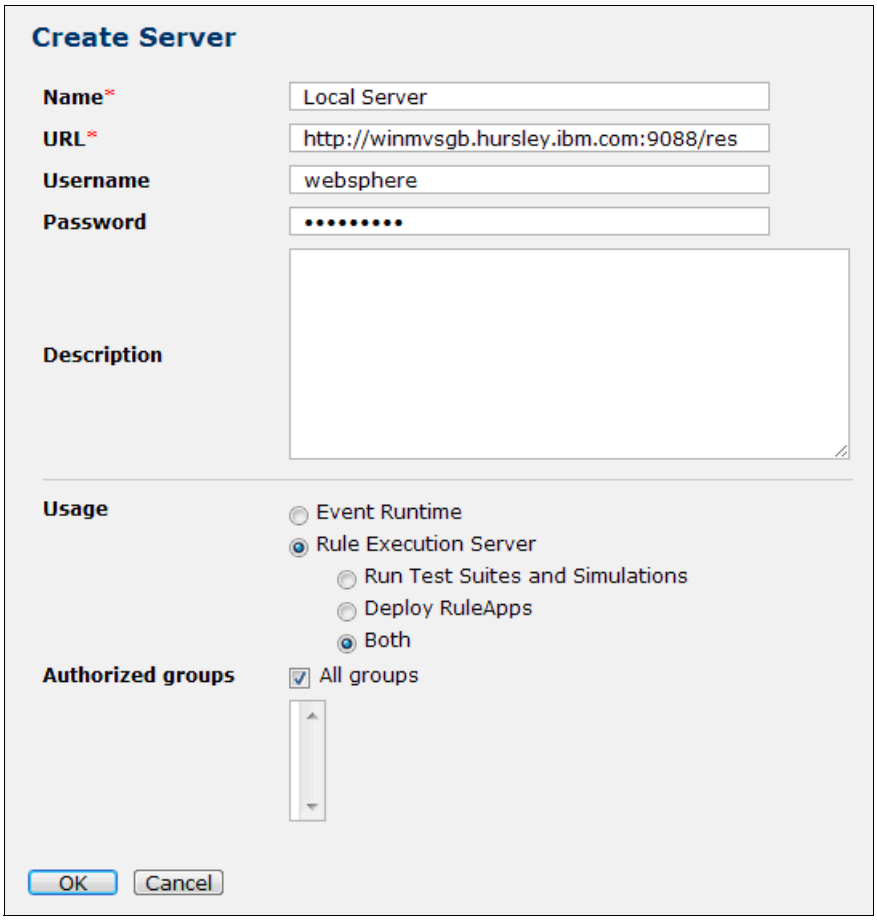


Figure 6-60 The Create Server page in Decision Center

7. The Manage Servers page opens again. Its content is updated with our new server information (Figure 6-61).

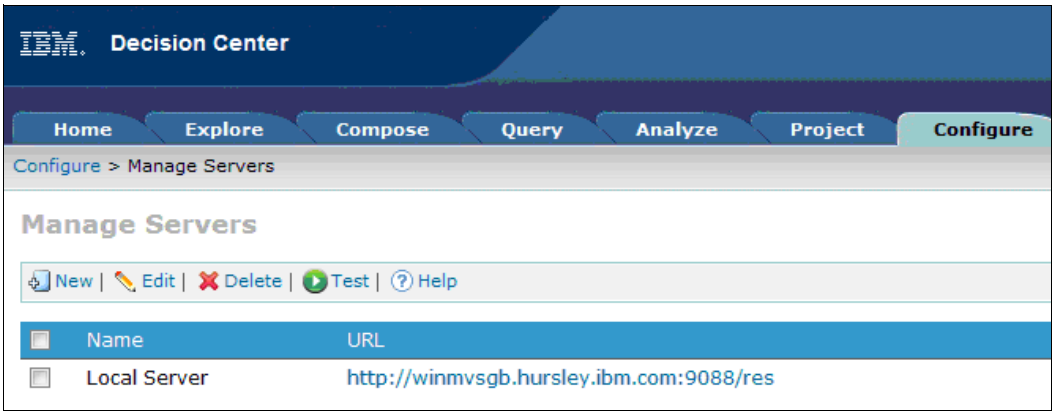


Figure 6-61 The Manage Servers page displaying information for a server definition

8. Now that a server definition is available to the business users, we ensure that all the scenario suite formats that we use to define scenario suites for the insurance-rules project are enabled. On Figure 6-61, click the **Project** tab to display the project menu that applies to the currently selected project (Figure 6-62).

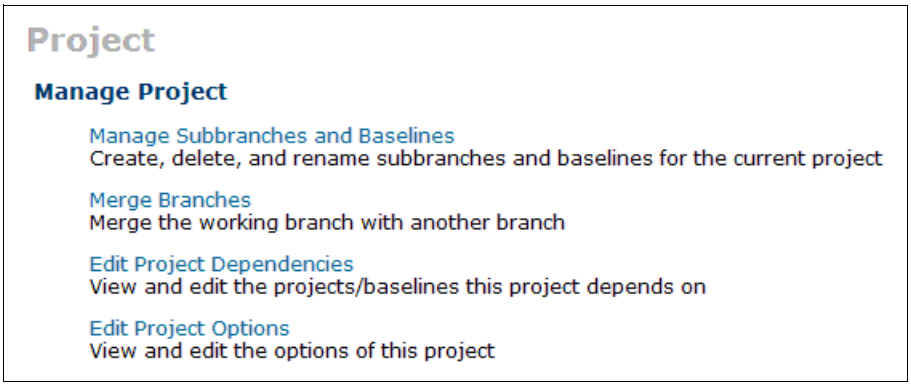


Figure 6-62 The Project menu in Decision Center

9. On the Manage Project page that is shown in Figure 6-62, click the **Edit Project Options** link so that we can edit the list of scenario suite formats available for the current project.

10. The Edit Project Options page opens (Figure 6-63).

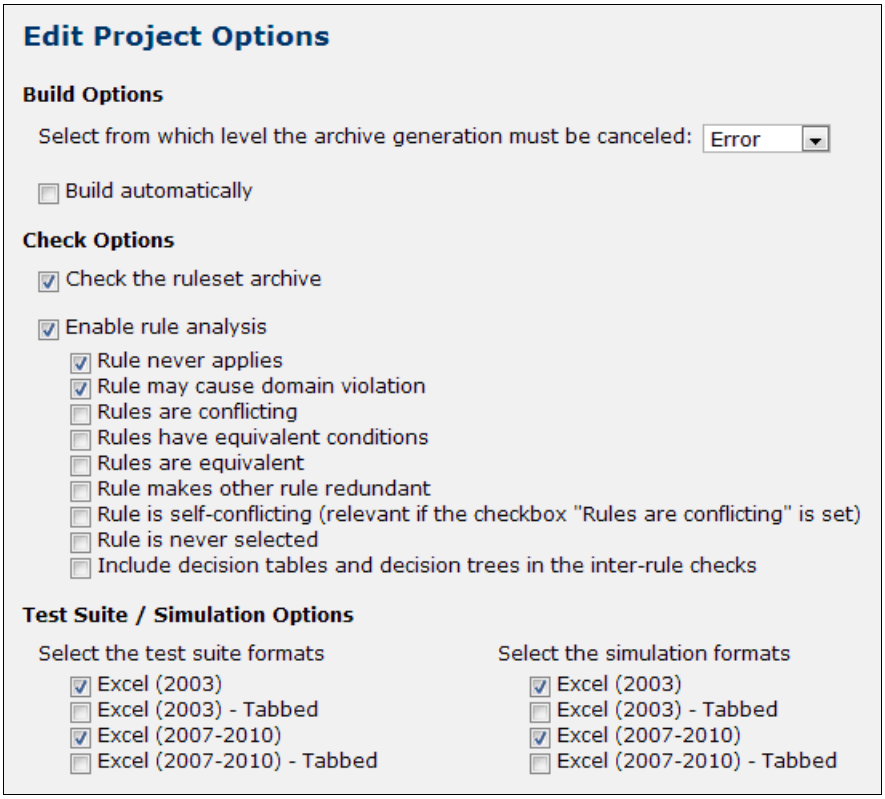


Figure 6-63 Edit Project Options in Decision Center

11. On the Edit Project Options page that is shown in Figure 6-63, in the Test Suite / Simulation Options section, select the test suite and simulation formats to make available for the current project. If you used the Excel (2003) format to create the test suite in Rule Designer, make sure that you enable it for the project in Decision Center. Only the Excel (2007-2010) format is enabled, by default. After you complete the format selection, click **OK** to save the changes.

Step 6: Create a test suite in Decision Center, run it, and display the execution report

Follow these steps to create a test suite in Decision Center, run it, and display the execution report:

1. Now that a server definition is created in Decision Center and the scenario suite formats to define test suites and simulation are enabled for the project, we create a test suite in Decision Center. Then, we run it and display its execution report.
2. We start by clicking the **Compose** tab, as shown in Figure 6-64 on page 170.

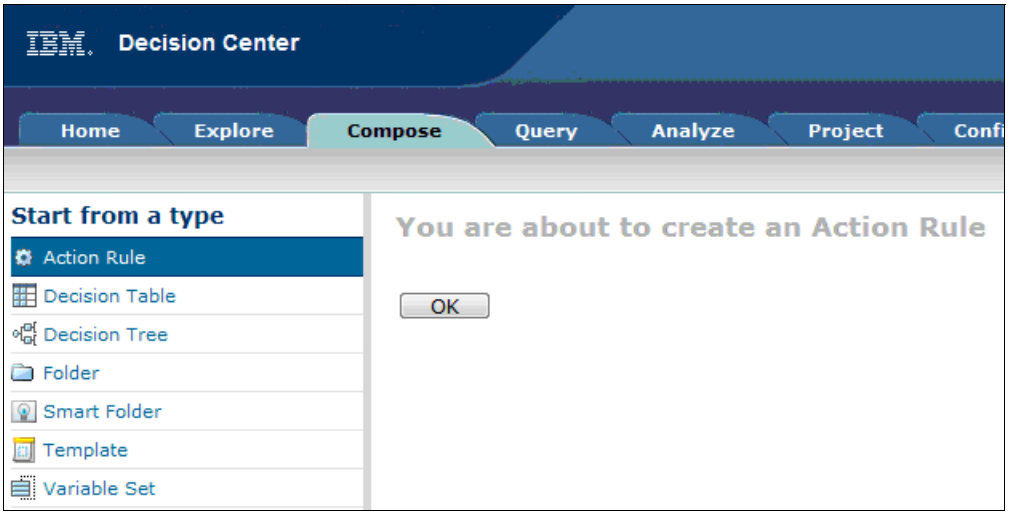


Figure 6-64 The Compose tab with the Action Rule type default selection

3. By default, the first time that we open the Compose page, Decision Center defaults to creating an Action Rule (Figure 6-64). To create a Test Suite instead, click **Test Suite** and the panel that is shown in Figure 6-65 appears.

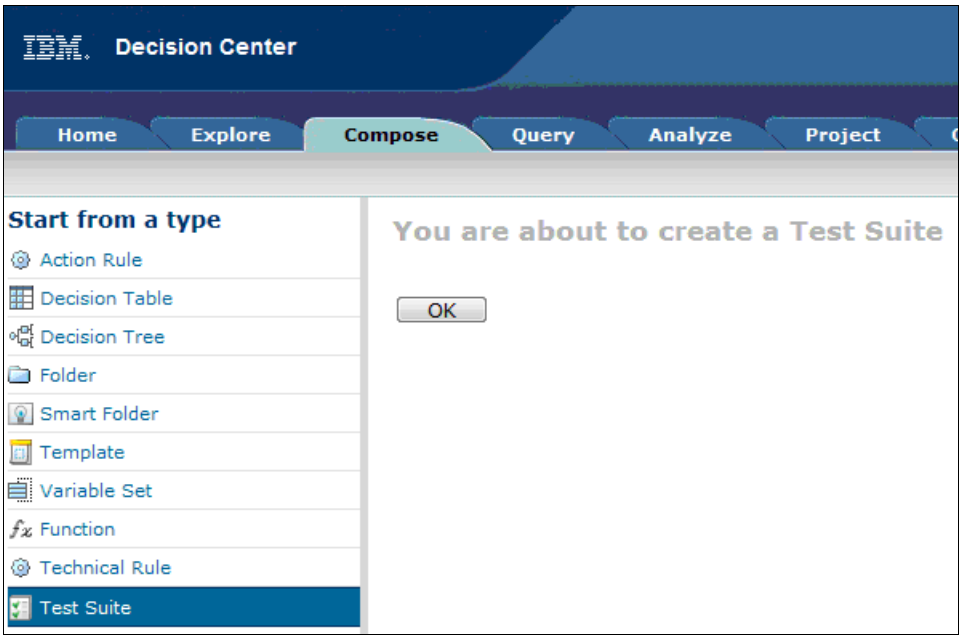


Figure 6-65 The Compose page with the Test Suite type selected

4. After we select Test Suite as the type of artifact to create, the right side of the page is updated. To start the new test suite wizard, click **OK** on the right side of the page (Figure 6-65).
5. Figure 6-66 on page 171 appears.

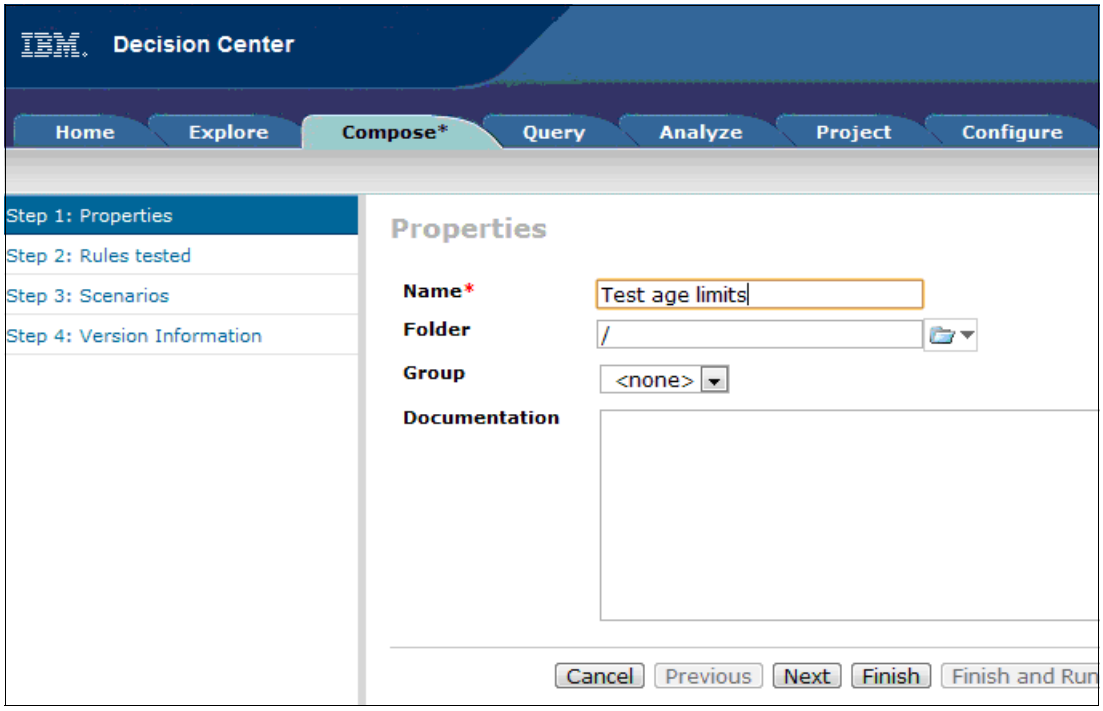


Figure 6-66 Test suite creation wizard in Decision Center: Properties

6. The first wizard page prompts us to enter the name, folder (location), group, and documentation for the new test suite. On the Properties page that is shown in Figure 6-66, enter the name `Test age limits`, leave the default values for the other fields, and click **Next**.
7. The Rules tested page opens for us to select rules to test (Figure 6-67).

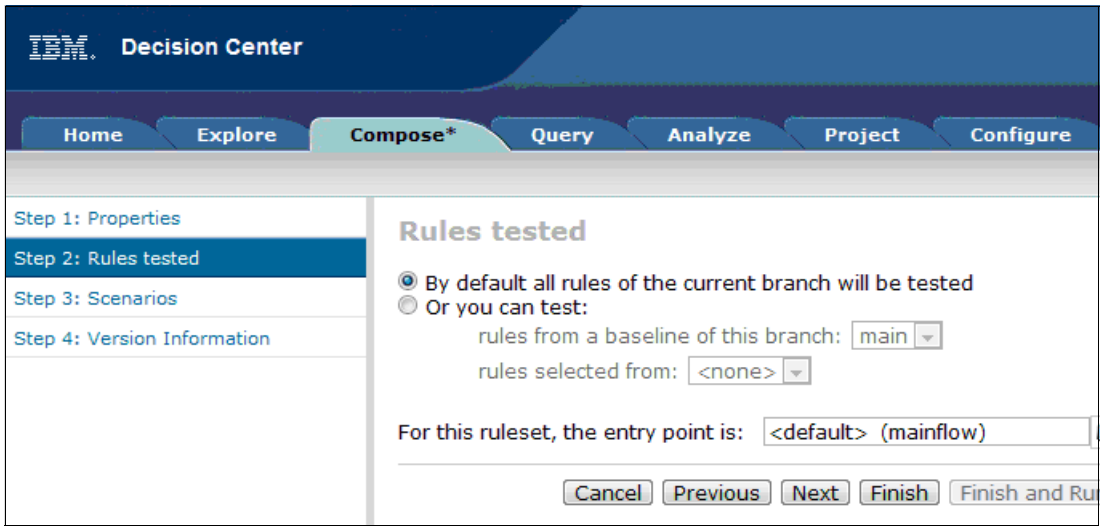


Figure 6-67 Test suite creation wizard in Decision Center: Rules tested

8. On the Rules tested page that is shown in Figure 6-67, you can choose to test the ruleset in its current status or select another version of the ruleset from a baseline. Also, you can limit the testing to a subset of the rules and select the entry point for the ruleset. Leave the default options that are selected, and click **Next**.

9. In the third page of the wizard, we must select the scenario suite format to use. We select the scenario suite format that we used to create the test suite in Rule Designer, **Excel (2003)**. Click **Choose File** to upload the Excel test suite that was created with Rule Designer (Figure 6-68).

Figure 6-68 Test suite creation wizard in Decision Center: Scenarios

10. On the Scenarios page that is shown in Figure 6-68, when the Excel scenario file is uploaded, click **Finish and Run** to save the new test suite. The Run Test age limits page opens (Figure 6-69).

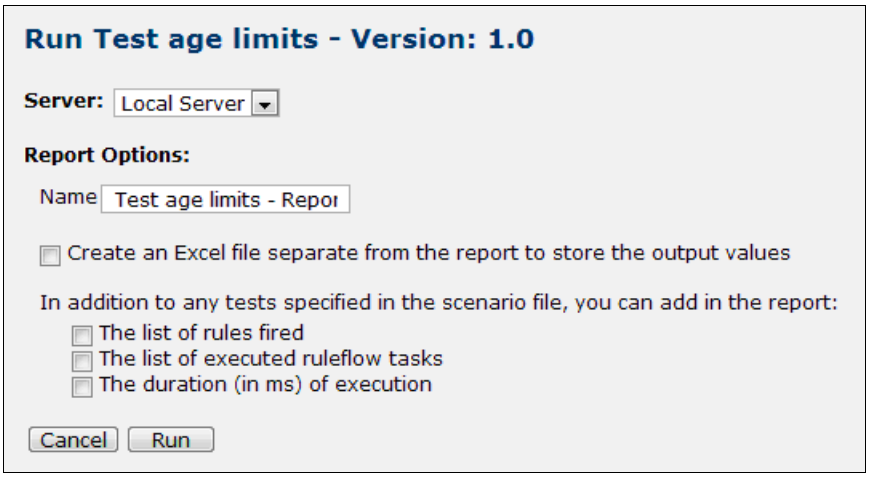


Figure 6-69 The Run Test suite page in Decision Center

11. To run the test suite, we click **Run**, and wait until the test report is displayed (Figure 6-70 on page 173).

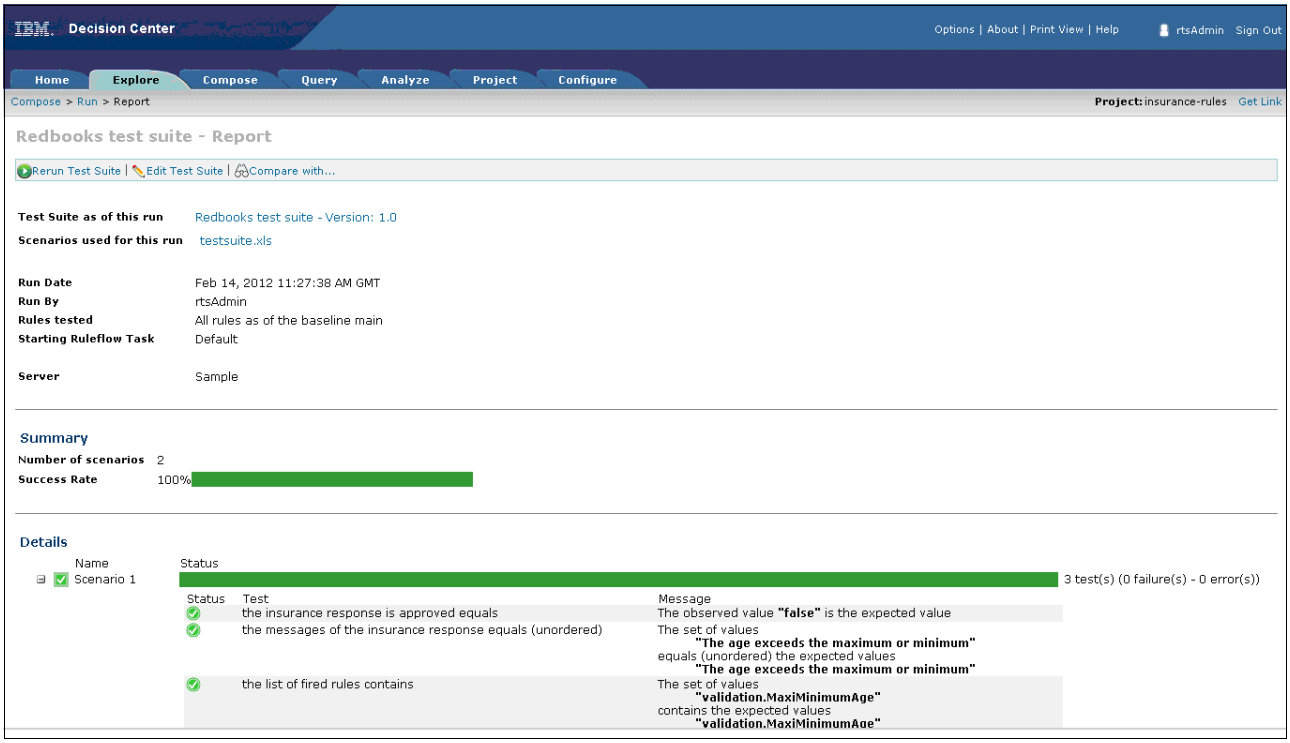


Figure 6-70 Test suite execution report in Decision Center

6.4.2 Scenario: Creating a custom scenario suite format to run simulations in Decision Center using input data from a VSAM file

This scenario demonstrates the following actions on a use case:

- Create a custom scenario suite format in Rule Designer
- Implement a scenario provider that reads input data from a VSAM file
- Add a KPI to the scenario suite format
- Add the scenario suite format to the customization, repackage it, and redeploy it to both the SSP and Decision Center
- Enable the new scenario suite format for the insurance-rules project in Decision Center
- Create and run a simulation with the VSAM scenario data format in Decision Center

A prerequisite to this scenario is to perform the steps of the scenario to import the insurance-rules project in Rule Designer, publish it to Decision Center, and create a DVS project with a customization.

Step 1: Create a custom scenario suite format in Rule Designer

Follow these steps to create a custom scenario suite format in Rule Designer:

1. To add a custom scenario suite format to the DVS project that we created in Decision Center, right-click **DVS Project** in the Rule Explorer. Then, select **Create a DVS artifact** → **Format** (Figure 6-71 on page 174).

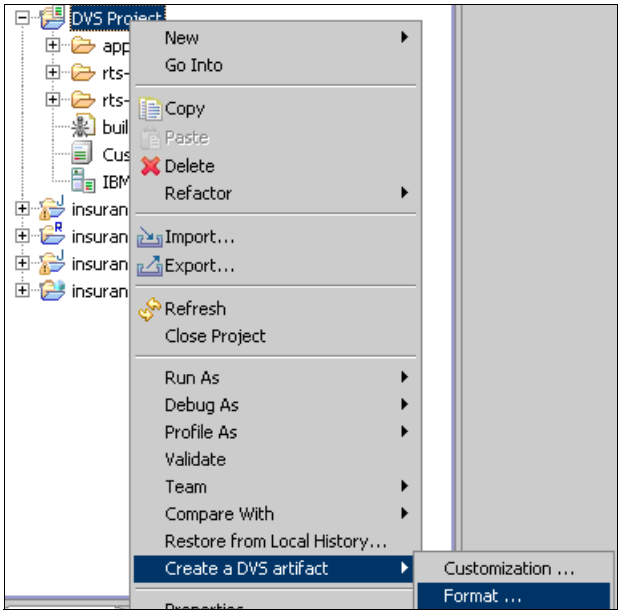


Figure 6-71 Creating a DVS artifact → Format for a DVS project

2. This action displays the first panel of the New DVS Format wizard. Keep the current selection, **DVS Project**, and click **Next >**. The new format is created in this DVS project (Figure 6-72).

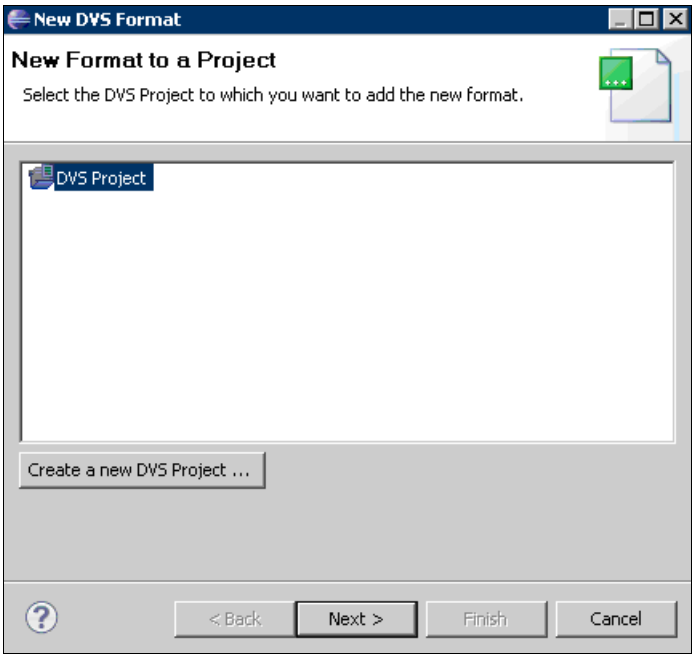


Figure 6-72 The first panel of the New DVS Format wizard

3. The second panel of the wizard is displayed. The Properties panel prompts us to enter a name for the new format and select a value for the precision for this format. The precision comparison is only implemented automatically if we later choose the class IlrExcel2003ScenarioProvider as the scenario provider for our format.

   If you create a custom scenario provider and want to use a precision comparison, implement the precision comparison yourself by retrieving the value entered here.

Enter the name `VSAM scenario data` for the new format and click **Next >** (Figure 6-73).

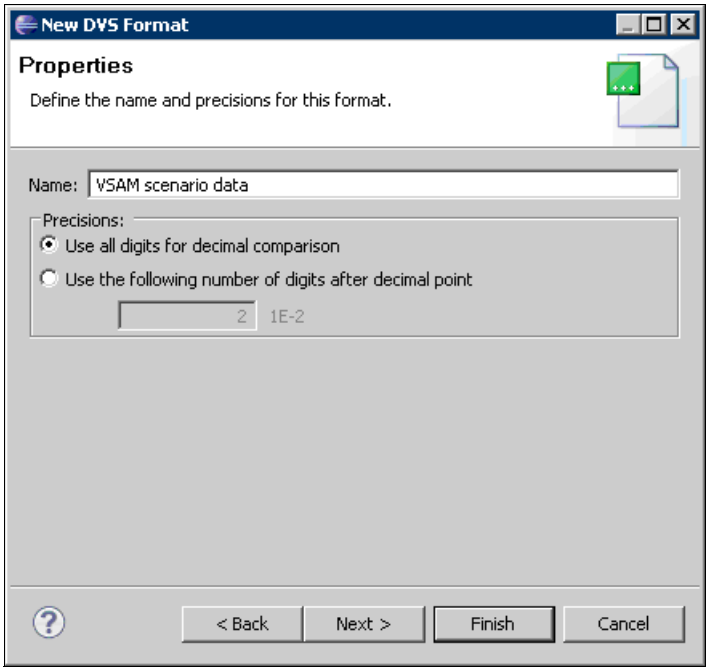


Figure 6-73 Properties panel of the New DVS Format wizard

4. The Scenario Provider panel opens, which prompts us to select the scenario provider implementation class to use for the wizard. We can optionally create a Scenario Provider (Figure 6-74). Because we want to create a custom scenario provider implementation that reads scenario data from a VSAM file, click the **Create a new Scenario Provider** link.

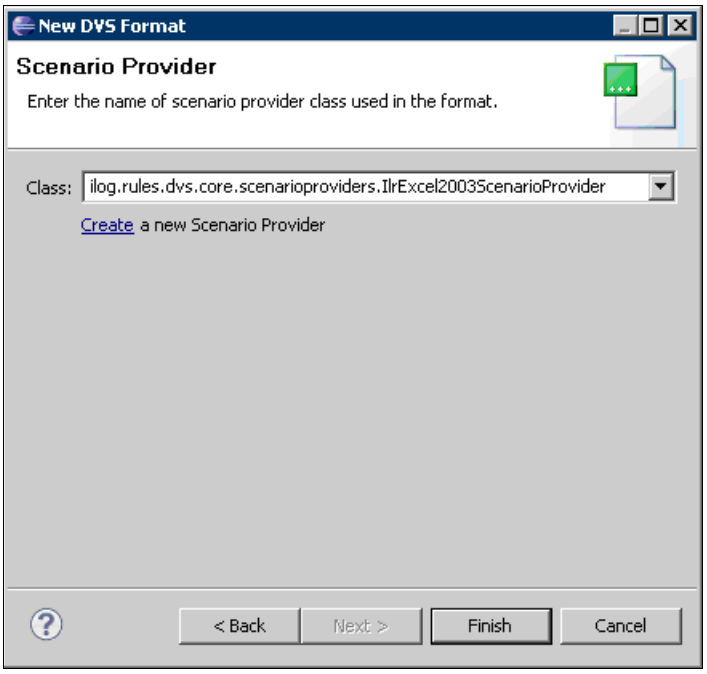


Figure 6-74 The Scenario Provider panel of the New DVS Format wizard

5. The New DVS Scenario Provider dialog opens. Enter the information about the new classes to create (Figure 6-75):
   - For the Package Name, enter `com.ibm.redbooks`.
   - For the Scenario Provider Class Name, enter `VSAMScenarioProvider`.
   - Ensure that **Generate the Decision Center renderer** is checked, and enter the class name `VSAMScenarioProviderResourcesRenderer`.
   - Ensure that the option **Generate the launch configuration plug-in project** is not selected.

   Click **OK** to close the dialog and display an updated version of the New DVS Format wizard panel.

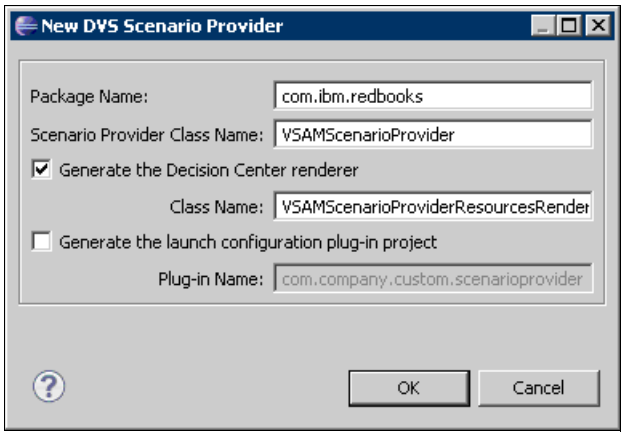


Figure 6-75 New DVS Scenario Provider dialog

6. The new scenario provider implementation class that we created is now selected as the scenario provider implementation class to use for the new scenario suite format (Figure 6-76 on page 177). Click **Finish**.

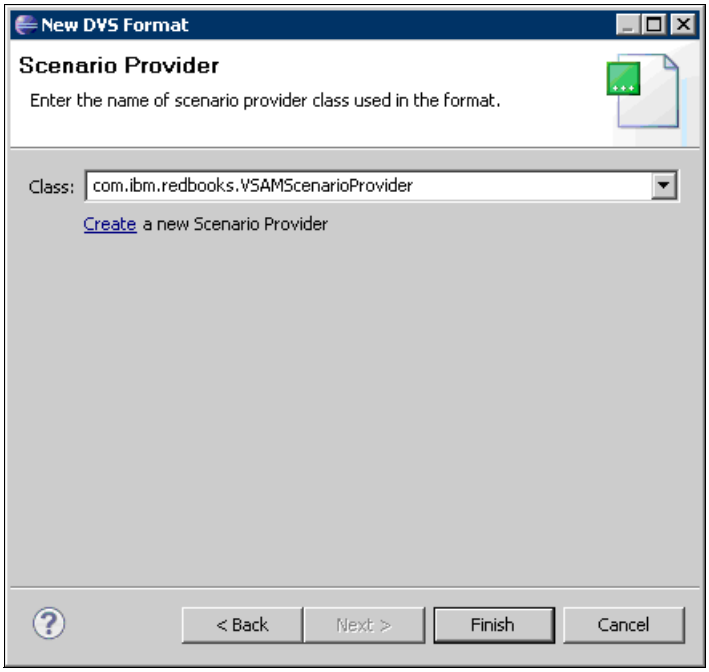


Figure 6-76 The updated Scenario Provider panel of the New DVS Format wizard

7. Rule Designer generates a scenario provider template and a resource renderer template in the source directory of the DVS project, along with the new format. The Format Editor opens with the definition of the new format (Figure 6-77).

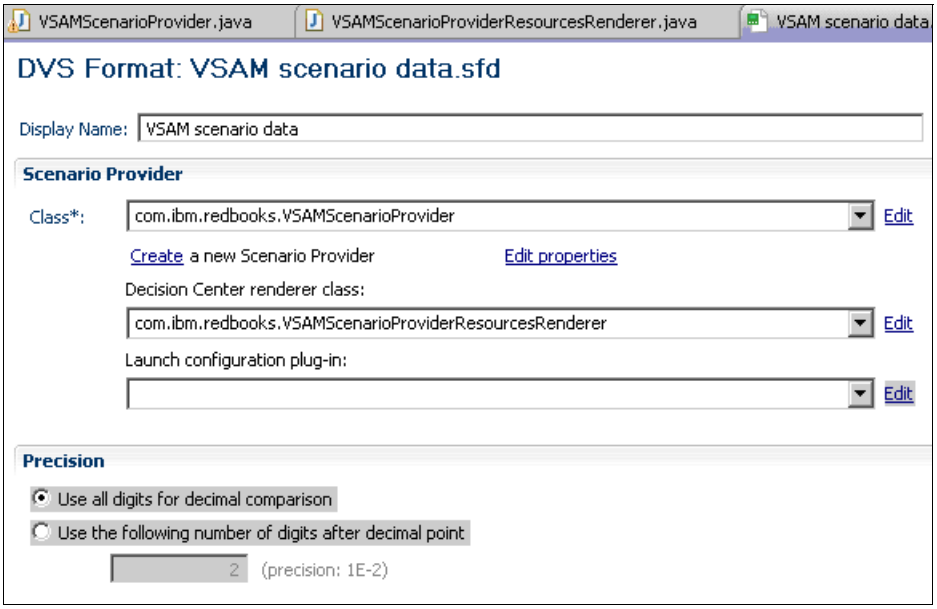


Figure 6-77 The scenario suite Format Editor in Rule Designer

Step 2: Implement a scenario provider that reads input data from a VSAM file

For this step, we create our own VSAM file and then populate it with sample data. The first class that we implement using the template that was created in the previous step is the VSAMScenarioProvider class. We select the **VSAMScenarioProvider.java** editor, to the left

of the scenario suite format editor. We list the details of the method that we implement for the VSAMScenarioProvider class (Figure 6-78):

- initialize(IlrScenarioSuiteExecutionContext context): This method is called by the SSP to initialize the scenario provider. The context parameter contains all information entered by the business user in Decision Center to define the simulation using the scenario suite format that was created, encoded as byte[] data. The initialize method retrieves this information and stores the values as instance attributes. We define the following attributes for this scenario provider:
  - The VSAM file to retrieve the scenario input data. This file is opened in the initialize method. Its name is retrieved in the context.
  - The index of the first VSAM record to use as the first scenario data. All records in the VSAM file are identified by a sequential eight-character key that contains the record index. The first record key is 00000001, the second key is 00000002, and so on.
  - The index of the last VSAM record to use as the last scenario data.
- close(): This method is called by the SSP at the end of the execution. It is in the implementation of this method that the VSAM file is closed.
- getScenarioCount(): This method is called by the SSP, after the initialization of the scenario provider, to retrieve the number of scenarios to run. The implementation calculates the number of scenarios to run using the values of the first and last record index without accessing the content of the VSAM file.
- getScenarioAt(int index): This method is called by the SSP for each scenario to run. The value of the index parameter for the first scenario is 0, and this value is incremented at each call. The implementation calculates the eight-character key of the VSAM record from the index parameter and retrieves it from the VSAM file. The implementation uses the marshaller classes generated for the insurance eligibility copybook to create Java input parameters for the ruleset, and it returns these Java input parameters in an IlrScenario instance.

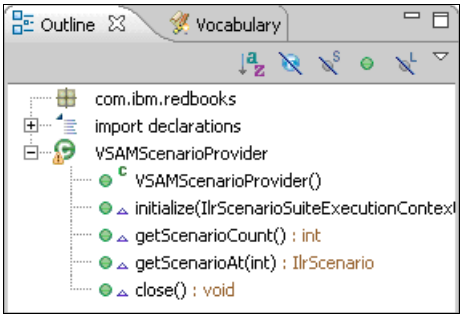


Figure 6-78 An outline of the VSAMScenarioProvider class to implement

The implementation uses the JZOS API to access the VSAM file on the z/OS runtime server during the execution of the scenario provider. As a consequence, we must set up the development environment to be able to compile code with the JZOS API inside Rule Designer. Follow these steps:

1. First, switch to the **Java perspective** by clicking the Java perspective icon in the upper-right corner of the Rule Designer window (Figure 6-79).

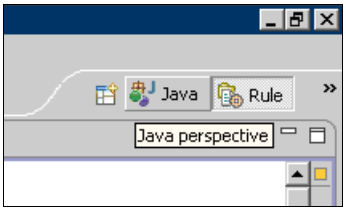


Figure 6-79 Switching to the Java perspective

2. From within the Package Explorer that is open in the Java perspective, right-click the **DVS Project** and select **Build Path** → **Add External Archives** (Figure 6-80).

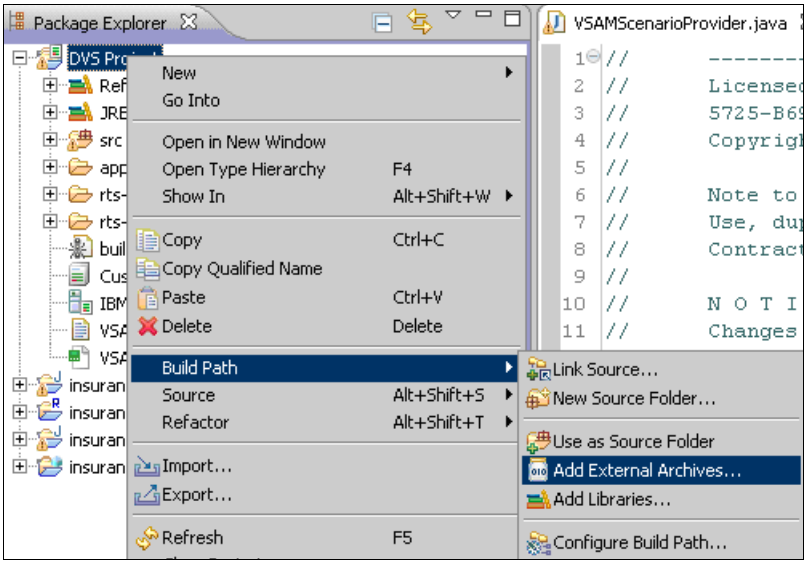


Figure 6-80 Build Path → Add External Archives

3. Using the file browser that is displayed after selecting the menu entry, locate the `ibmjzos.jar` file in the z file system directory `<JRE-INSTALL-ROOT>/lib/ext` and add it to the build path (Figure 6-81 on page 180).

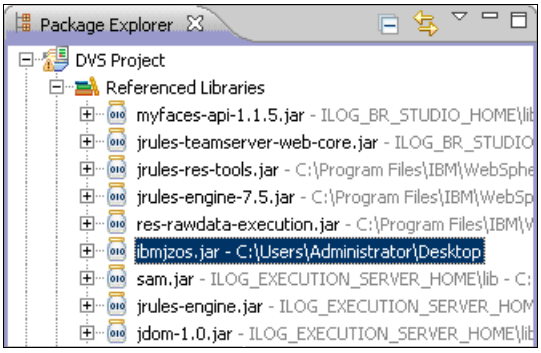


Figure 6-81 The ibmjzos.jar external jar successfully added to the build path

4. Now that the JZOS API is available in the DVS project, update the content of the `VSAMScenarioProvider.java` Source Code Editor with the content that is shown in Example 6-1.

Example 6-1 Update the content of the VSAMScenarioProvider.java Source Code Editor

```
package com.ibm.redbooks;
import ilog.rules.dvs.common.input.IlrScenarioSuiteDescriptor;
import ilog.rules.dvs.common.output.IlrScenarioProviderException;
import ilog.rules.dvs.common.output.IlrTestingException;
import ilog.rules.dvs.core.IlrInitializationException;
import ilog.rules.dvs.core.IlrScenario;
import ilog.rules.dvs.core.IlrScenarioProvider;
import ilog.rules.dvs.core.IlrScenarioSuiteExecutionContext;
import ilog.rules.dvs.core.impl.IlrScenarioImpl;
import java.io.Serializable;
import java.text.NumberFormat;
import java.util.HashMap;
import java.util.Map;
import marshaller.RequestMarshaller;
import xom.Request;
import xom.Response;
import com.ibm.jzos.ZFile;
import com.ibm.jzos.ZFileException;
public class VSAMScenarioProvider implements IlrScenarioProvider, Serializable
{
   // A serial number for our serializable class
   private static final long serialVersionUID = 322801542385902272L;
   // The VSAM_FILE_NAME parameter to initialize the scenario provider
   public static final String VSAM_FILE_NAME = "VSAM_FILE_NAME";
   // The FIRST_INDEX parameter to initialize the scenario provider
   public static final String FIRST_INDEX = "FIRST_INDEX";
   // The LAST_INDEX parameter to initialize the scenario provider
   public static final String LAST_INDEX = "LAST_INDEX";
   // The name of the input parameter for the ruleset
   public static final String REQUEST_IN_PARAMETER = "request";
   // The name of the output parameter for the ruleset
   public static final String RESPONSE_OUT_PARAMETER = "response";
   // The index of the first scenario in the VSAM file
   private int firstIndex;
   // The index of the last scenario in the VSAM file
   private int lastIndex;
```

```
// The VSAM file in which the scenarios are stored
private ZFile vsamFile = null;
// The record length in the VSAM file
private int lrecl = 138;
// The record key length in the VSAM file
private int keyLen = 8;
// A format instance we use to translate a scenario index into
// a record key in the VSAM file
private NumberFormat format = null;
/**
 * Create a new instance.
 */
public VSAMScenarioProvider() {
   super();
}
/**
 * Callback method invoked by the runner in order to inject
 * the execution context into the component.
 */
public void initialize(IlrScenarioSuiteExecutionContext context)
      throws IlrInitializationException {
   IlrScenarioSuiteDescriptor descriptor =
               context.getScenarioSuiteDescriptor();
   ////////////////////////////////////////////////////
   // Retrieve the scenario suite parameters (name of
   // the VSAM file and range of scenario indexes) from
   // the context.
   // When the parameter values are entered a string in
   // Decision Center, they are then encoded as byte[],
   // so this is the type of parameters we are looking for.
   ////////////////////////////////////////////////////
   Object encodedFirstIndex = descriptor.get(FIRST_INDEX);
   if (encodedFirstIndex != null &&
      encodedFirstIndex instanceof byte[]) {
      this.firstIndex = Integer.valueOf(new String(
            (byte[]) encodedFirstIndex));
   } else {
      throw new IlrInitializationException(
            "Cannot find a suitable value in context for "
                  + FIRST_INDEX);
   }
   Object encodedLastIndex = descriptor.get(LAST_INDEX);
   if (encodedLastIndex != null &&
      encodedLastIndex instanceof byte[]) {
      this.lastIndex = Integer.valueOf(new String(
            (byte[]) encodedLastIndex));
   } else {
      throw new IlrInitializationException(
            "Cannot find a suitable value in context for "
               + LAST_INDEX);
   }
   // Check constraint on indexes
   if (this.lastIndex < this.firstIndex) {
      throw new IlrInitializationException(LAST_INDEX
            + " should be lower or equal to "
```

```
            + FIRST_INDEX);
      }
      ////////////////////////////////////////////////////
      // Retrieve the name of the VSAM file from the context
      // and open it. In the context, the name of the VSAM file
      // is stored as a byte[].
      ////////////////////////////////////////////////////
      String vsamFileName;
      Object encodedVSAMFileName = descriptor.get(VSAM_FILE_NAME);
      if (encodedVSAMFileName != null &&
         encodedVSAMFileName instanceof byte[]) {
         vsamFileName = new String((byte[]) encodedVSAMFileName);
      } else {
         throw new IlrInitializationException(
               "Cannot find a suitable value in context for "
                     + VSAM_FILE_NAME);
      }
      // The VSAM file name must start with //
      if (!vsamFileName.startsWith("//")) {
         vsamFileName = "//" + vsamFileName;
      }
      try {
         // Check that the file exists
         if (!ZFile.exists(vsamFileName)) {
            throw new IlrInitializationException("VSAM File "
                  + vsamFileName + " does not exist");
         }
         // Open the VSAM file as read only
         this.vsamFile = new ZFile(vsamFileName, "rb,type=record");
      } catch (ZFileException e) {
         throw new IlrInitializationException(e);
      }
      ////////////////////////////////////////////////////
      // Initialize the number formatter we use to create
      // VSAM record keys from the scenario indexes.
      ////////////////////////////////////////////////////
      this.format = NumberFormat.getIntegerInstance();
      this.format.setMinimumIntegerDigits(8);
      this.format.setMaximumIntegerDigits(8);
      this.format.setGroupingUsed(false);
   }
   /**
    * Return the count of scenarios for this provider.
    *
    * @return the count of scenarios for this provider
    * @throws IlrTestingException
    *             if an error occurred while computing the scenarios count
    */
   public int getScenarioCount() throws IlrScenarioProviderException {
      return (lastIndex - firstIndex) + 1;
   }
   /**
    * Return the scenario at specified index
    *
    * @param indx
```

```
 *            the index of the scenario
 * @return the scenario at specified index
 * @throws IlrTestingException
 *             if an exception occurred while retrieving the scenario
 */
public IlrScenario getScenarioAt(int indx)
      throws IlrScenarioProviderException {
   try {
      ///////////////////////////////////
      // Compute the key of the VSAM record
      ///////////////////////////////////
      String keyAsString = this.format.format(
            this.firstIndex + indx);
      ///////////////////////////////////
      // Read the record from the VSAM file
      ///////////////////////////////////
      byte[] record = new byte[lrecl];
      byte[] keyBytes = keyAsString
            .getBytes(ZFile.DEFAULT_EBCDIC_CODE_PAGE);
      boolean located = this.vsamFile.locate(keyBytes, 0, keyLen,
            ZFile.LOCATE_KEY_EQ);
      if (!located) {
         throw new IlrScenarioProviderException(
               "Cannot find record for key "
               + keyAsString
               + " in the VSAM file");
      }
      this.vsamFile.read(record);
      ///////////////////////////////////
      // Create a Request java object from the record
      ///////////////////////////////////
      RequestMarshaller requestMarshaller =
         new RequestMarshaller();
      // The key value is not part of the COBOL input parameter
      // object, so we skip it in the record
      Request requestInputParameter = (Request) requestMarshaller
            .unmarshal(new HashMap<String, Integer>(),
                  record, keyLen);
      ///////////////////////////////////
      // Create the map of input parameters for
      // the ruleset execution
      ///////////////////////////////////
      Map<String, Object> scenarioInputParameters =
         new HashMap<String, Object>();
      scenarioInputParameters.put(REQUEST_IN_PARAMETER,
            requestInputParameter);
      scenarioInputParameters.put(RESPONSE_OUT_PARAMETER,
            new Response());
      ///////////////////////////////////
      // Create and return the scenario itself
      ///////////////////////////////////
      IlrScenarioImpl returnedValue = new IlrScenarioImpl();
      returnedValue.setName("Scenario " + indx);
      returnedValue.setInputParameters(scenarioInputParameters);
      return returnedValue;
```

```
      } catch (Throwable t) {
         throw new IlrScenarioProviderException(t);
      }
   }
   /**
    * Close the scenario provider.
    */
   public void close() {
      if (this.vsamFile != null) {
         try {
            this.vsamFile.close();
         } catch (ZFileException e) {
            // Nothing can be done here...

         }
      }
   }
}
```

We implement the second class, VSAMScenarioProviderResourceRenderer, which is the resource renderer for Decision Center. The renderer displays an HTML form to be used by a business user to define values for the initialization parameters of the scenario provider. The renderer encodes these parameters as byte[] so that they are stored in the execution context that the scenario provider instance uses at initialization time. The renderer also provides an HTML rendering of the list of initialization parameters to display in the execution report and the various simulation views that are available in Decision Center.

The implementation of the resource renderer uses the JavaServer Faces (JSF) API to display information in the Decision Center user interface. We implement the following methods:

- decode(IlrScenarioFormatDescriptor formatDescriptor, Map<String, byte[]> resources, FacesContext context, UIComponent component): This method stores the values of the scenario provider initialization parameters as byte[]. The original values that were entered by the business user are retrieved from the request parameters that are stored in the FacesContext instance that is provided as a parameter.
- encodeAsEditor(IlrScenarioFormatDescriptor formatDescriptor, Map<String, byte[]> resources, FacesContext context, UIComponent component): This method displays an HTML form for the business user to enter the initialization parameters for the scenario provider.
- encodeAsViewer(IlrScenarioFormatDescriptor formatDescriptor, Map<String, byte[]> resources, FacesContext context, UIComponent component): This method outputs the initialization parameters of the scenario provider in an HTML page.

Example 6-2 shows the source code to update the content of the `VSAMScenarioProviderResourceRender.java` Source Code Editor.

Example 6-2 Source code to update the VSAMScenarioProviderResourceRender.java file

```
package com.ibm.redbooks;
import ilog.rules.dvs.common.descriptors.IlrScenarioFormatDescriptor;
import
ilog.rules.teamserver.web.components.renderers.IlrScenarioSuiteResourcesRenderer;
import java.io.IOException;
import java.util.Map;
import javax.faces.component.UIComponent;
import javax.faces.context.ExternalContext;
```

```
import javax.faces.context.FacesContext;
import javax.faces.context.ResponseWriter;
public class VSAMScenarioProviderResourcesRenderer implements
      IlrScenarioSuiteResourcesRenderer {
   /**
    * Decode any new state of the specified resources from the request
    * contained in the specified {@link FacesContext}, validate the state,
    * and store that state in the resources Map.
    *
    * @param formatDescriptor
    *            the format description
    * @param resources
    *            the resources to be decoded
    * @param context
    *            {@link FacesContext} for the request we are processing
    * @param component
    *            the current {@link UIComponent}
    *
    */
   public void decode(IlrScenarioFormatDescriptor formatDescriptor,
         Map<String, byte[]> resources, FacesContext context,
         UIComponent component) {
      // Retrieve the request parameters from the context
      ExternalContext extCtx = context.getExternalContext();
      Map<?,?> requestParameterMap = extCtx.getRequestParameterMap();
      // Populate the resources map with byte[] values for
      // the initialization parameters retrieved in the request
      addResourceFromRequestParameter(requestParameterMap, resources,
            VSAMScenarioProvider.VSAM_FILE_NAME, context,
            component);
      addResourceFromRequestParameter(requestParameterMap, resources,
            VSAMScenarioProvider.FIRST_INDEX, context,
            component);
      addResourceFromRequestParameter(requestParameterMap, resources,
            VSAMScenarioProvider.LAST_INDEX, context,
            component);
   }
   /**
    * Render as an editor the current state of the specified resources.
    *
    * @param formatDesc
    *            the format description
    * @param resources
    *            the resources to be rendered
    * @param context
    *            {@link FacesContext} for the request we are processing
    * @param component
    *            the current {@link UIComponent}
    * @throws IOException
    *             if an input/output error occurs
    */
   public void encodeAsEditor(IlrScenarioFormatDescriptor formatDesc,
         Map<String, byte[]> resources, FacesContext context,
         UIComponent component) throws IOException {
      // Retrieve a writer to the HTML page
```

```
    ResponseWriter writer = context.getResponseWriter();
    writer.writeAttribute("size", 30, null);
    writer.endElement("input");
    // Add test fields to input the scenario provider initialization
    // parameters
    addTextFieldInEditor(writer, "VSAM File: ",
          VSAMScenarioProvider.VSAM_FILE_NAME, resources,
          component, context);
    addTextFieldInEditor(writer, "Index of the first scenario: ",
          VSAMScenarioProvider.FIRST_INDEX, resources,
          component, context);
    addTextFieldInEditor(writer, "Index of the last scenario: ",
          VSAMScenarioProvider.LAST_INDEX, resources,
          component, context);
}
/**
 * Render as a viewer the current state of the specified resources.
 *
 * @param formatDesc
 *            the format description
 * @param resources
 *            the resources to be rendered
 * @param context
 *            {@link FacesContext} for the request we are processing
 * @param component
 *            the current {@link UIComponent}
 * @throws IOException
 *             if an input/output error occurs
 */
public void encodeAsViewer(IlrScenarioFormatDescriptor formatDesc,
      Map<String, byte[]> resources, FacesContext context,
      UIComponent component) throws IOException {
    // Retrieve a writer to the HTML page
    ResponseWriter writer = context.getResponseWriter();
    // Output the scenario provider initialization parameters
    // as a comma-separated list of key: value
    writer.writeText(" VSAM File: ", null);
    byte[] c = resources.get(VSAMScenarioProvider.VSAM_FILE_NAME);
    writer.writeText(new String(c), null);
    writer.writeText(", First scenario index: ", null);
    c = resources.get(VSAMScenarioProvider.FIRST_INDEX);
    writer.writeText(new String(c), null);
    writer.writeText(", Last scenario index: ", null);
    c = resources.get(VSAMScenarioProvider.LAST_INDEX);
    writer.writeText(new String(c), null);
}

/**
 * Private method to ask a text field in the HTML page
 * to input the value of a scenario provider initialization
 * parameter.
 *
 * @param writer writer to the HTML page
 * @param textFieldLabel the label for the initialization
 * parameter text field
```

```
     * @param resourceKey the key of the initialization parameter
     * in the resources map
     * @param resources the resources map
     * @param component the UI component for the HTML page output
     * @param context the Faces context
     * @throws IOException
     */
    private void addTextFieldInEditor(ResponseWriter writer,
          String textFieldLabel,
          String resourceKey,
          Map<String, byte[]> resources,
          UIComponent component,
          FacesContext context) throws IOException {
        // output the label for the field
        writer.writeText(textFieldLabel + ": ", null);
        // output the text field
        writer.startElement("input", component);
        writer.writeAttribute("type", "text", null);
        writer.writeAttribute("name", component.getClientId(context)
              + "_" + resourceKey, null);
        if (resources.get(resourceKey) != null) {
           byte[] c = resources.get(resourceKey);
           writer.writeAttribute("value", new String(c), null);
        }
        writer.writeAttribute("size", 30, null);
        // this submits the form as soon as something is added inside,
        // so the Finish and run button are enabled in the new
        // simulation wizard page.
        writer.writeAttribute("onchange", "this.form.submit();", null);
        writer.endElement("input");
        writer.startElement("br", component);
        writer.endElement("br");
    }

    /**
     * Add the byte[] value of a scenario provider initialization
     * parameter in the resources map.
     * @param params the request parameters
     * @param resources the resources map
     * @param resourceKey the key for the resource to add
     * @param context the Faces context
     * @param component the UI component
     */
    protected void addResourceFromRequestParameter(Map<?,?> params,
          Map<String, byte[]> resources, String resourceKey,
          FacesContext context, UIComponent component) {
        String count = (String)params.get(component.getClientId(context)
              + "_" + resourceKey);
        resources.put(resourceKey, count.getBytes());
    }
}
```

Step 3: Add a KPI to the scenario suite format

Now that the scenario site provider and the associated renderer are defined for the scenario suite format, the next step is to add a KPI to use for running simulations against the insurance-rules ruleset with this format.

Follow these steps:

1. Back at the Format Editor, click **New** in the KPI section (Figure 6-82).

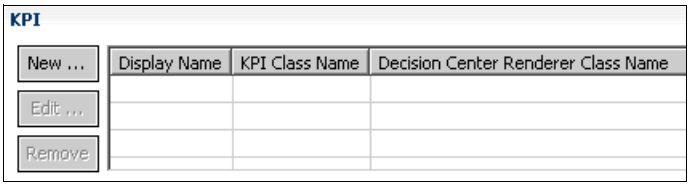


Figure 6-82 The KPI section of the scenario suite Format Editor

2. This action opens the KPI Definition dialog. In this dialog, enter the KPI Display Name `Number of requests approved` for the new KPI (Figure 6-83). After entering the KPI display name, click the **Create** link (on the right) to display the New KPI Generation dialog.

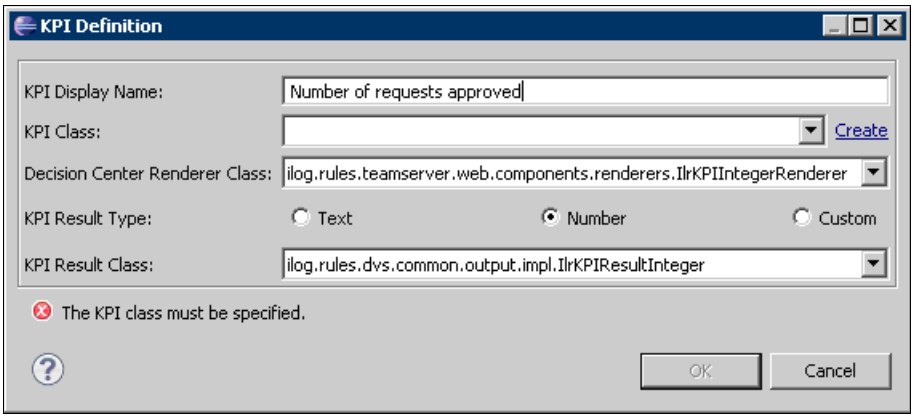


Figure 6-83 KPI Definition dialog

3. Enter the following information for the new KPI class that we generate (Figure 6-84 on page 189):
   - For Package, type `com.ibm.redbooks`.
   - For KPI Class, type `ApprovedRequestsCountKPI`.
   - For Decision Center Renderer Class, leave the default value. We select a generic KPI renderer later.
   - For KPI Result Type, select **Number**.

   On the New KPI Generation page that is shown in Figure 6-84 on page 189, click **OK** to trigger the generation of two templates for the custom KPI class and the associated renderer.

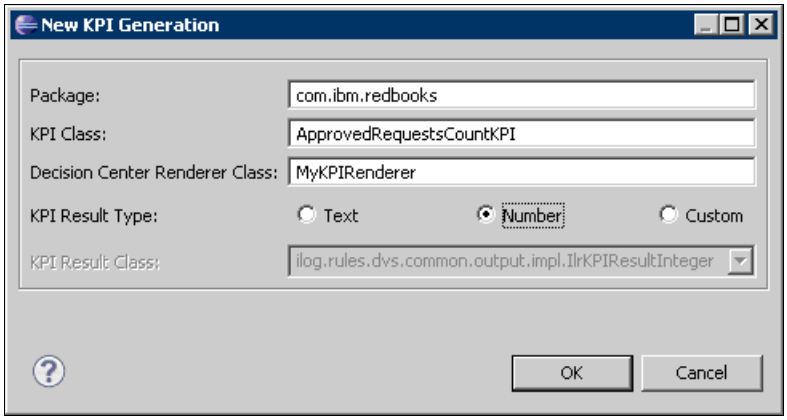


Figure 6-84 New KPI Generation dialog

The New KPI Generation page closes, and the content of the KPI Definition dialog is updated to reflect the information that we entered.

4. In the KPI Definition window that is shown in Figure 6-85, replace the com.ibm.redbooks.MyKPIRenderer Decision Center Renderer Class selection with the **ilog.rules.teamserver.web.components.renderers.IlrKPIIntegerRenderer** default renderer to avoid having to implement a custom KPI renderer. Then, click **OK** (Figure 6-85).

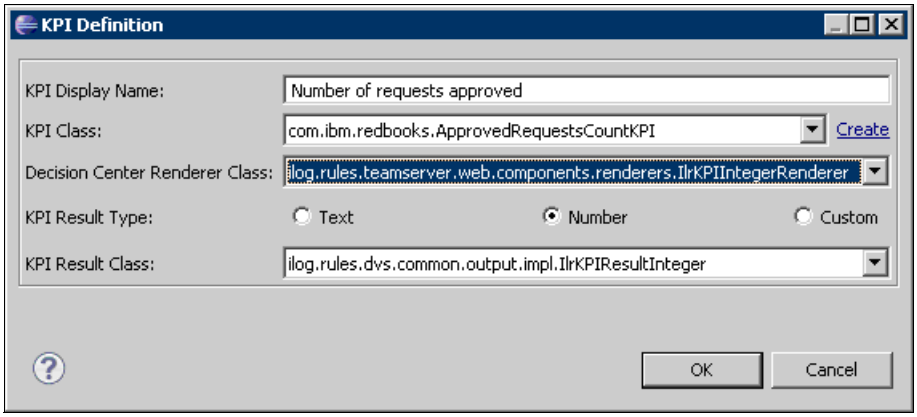


Figure 6-85 Selection of a default KPI renderer class for a KPI definition

5. The Package Explorer shows that the two class templates, ApprovedRequestsCountKPI.java and MyKPIRenderer.java, are generated in the DVS Project. Because we are not going to implement MyKPIRenderer, delete it by right-clicking it and selecting **Delete** (Figure 6-86 on page 190).

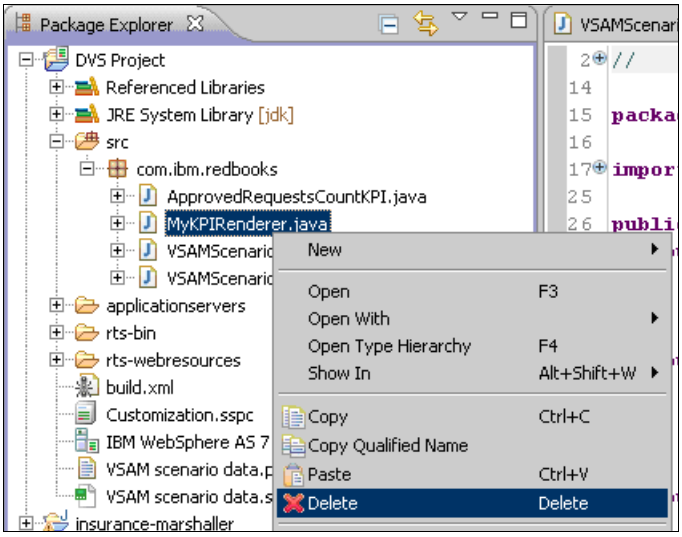


Figure 6-86 Deleting a Java class

6. Next, we implement the ApprovedRequestCountKPI class (Figure 6-87 on page 191), so that it calculates the number of approved requests during the execution of a simulation. We implement the following methods for this class:
   - initialize(IlrScenarioSuiteExecutionContext context): This method is called by the SSP to initialize the KPI instance before the execution of a simulation. Because this KPI does not perform any initialization before a simulation begins its execution, this method is empty.
   - close(): This method is called by the SSP after the execution of a simulation, so that the KPI instance can release any resources that it used during the execution of the simulation. Because this KPI does not have any resources to release at the end of a simulation, this method is empty.
   - onScenarioBegin(IlrScenario scenario, IlrSessionRequest request): This method is called by the SSP before the execution of one scenario of the simulation. The input parameters for this method are the scenario to be executed, and the request to send to the rule engine for the execution of the scenario (rule session API). This KPI implementation performs a calculation on the result of an execution only, so this method is empty.
   - onScenarioEnd(IlrScenario scenario, IlrSessionRequest request, IlrSessionResponse response): This method is called by the SSP after the execution of one scenario of the simulation. The input parameter response contains the rule engine response to the initial execution request. This response contains the output parameters resulting from the execution, which this KPI implementation uses to calculate its value.
   - getKPIResult(): This method is called by the SSP at the end of the execution of the simulation to retrieve the value of the KPI. This implementation returns an integer value that is wrapped into an IlrKPIResultInteger instance.

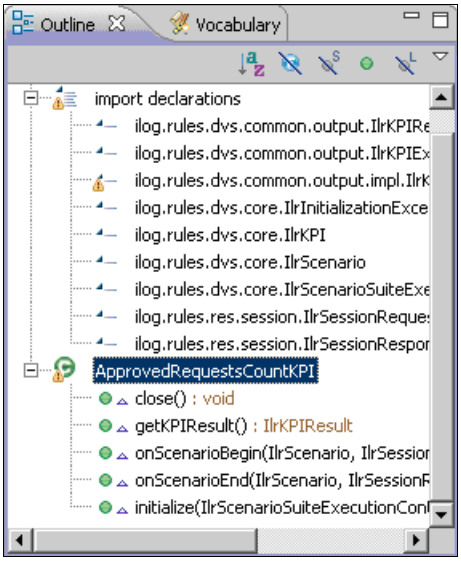


Figure 6-87 An outline of the source code template for ApprovedRequestsCountKPI

7. On the Outline tab in the Package Explorer that is shown in Figure 6-87, double-click the **ApprovedRequestsCountKPI** class to display the content of the class source code template in the Java Source Code Editor. Replace its content with the text that is shown in Example 6-3.

Example 6-3 Source code for ApprovedRequestsCountKPI.java

```
package com.ibm.redbooks;
import ilog.rules.dvs.common.output.IlrKPIException;
import ilog.rules.dvs.common.output.IlrKPIResult;
import ilog.rules.dvs.common.output.impl.IlrKPIResultInteger;
import ilog.rules.dvs.core.IlrInitializationException;
import ilog.rules.dvs.core.IlrKPI;
import ilog.rules.dvs.core.IlrScenario;
import ilog.rules.dvs.core.IlrScenarioSuiteExecutionContext;
import ilog.rules.res.session.IlrSessionRequest;
import ilog.rules.res.session.IlrSessionResponse;
import java.util.Map;
import xom.Response;
public class ApprovedRequestsCountKPI implements IlrKPI {
   // The KPI value: the number of requests approved during
   // the execution of the simulation
   private int numberOfRequestsApproved = 0;
   public void close() {
      // No implementation
   }
   public IlrKPIResult getKPIResult() throws IlrKPIException {
      // Return the number of requests approved
      IlrKPIResultInteger returnedValue = new IlrKPIResultInteger();
      returnedValue.setValue(this.numberOfRequestsApproved);
      return returnedValue;
   }
   public void onScenarioBegin(IlrScenario scenario,
         IlrSessionRequest request)
         throws IlrKPIException {
```

```
      // No implementation
   }
   public void onScenarioEnd(IlrScenario scenario,
         IlrSessionRequest request,
         IlrSessionResponse response) throws IlrKPIException {
      try {
         // Retrieve the output parameter of the ruleset
         Map<String, Object> outputs =
            response.getOutputParameters();
         Response r =
            (Response) outputs.get("response");
         if (r.getApproved()) {
            // Increment the KPI value if the request
            // was approved
            this.numberOfRequestsApproved++;
         }
      } catch (Throwable t) {
         throw new IlrKPIException(t);
      }
   }
   public void initialize(IlrScenarioSuiteExecutionContext context)
         throws IlrInitializationException {
      // No implementation
   }
}
```

8. Before moving to the next step, make sure that you save the content of the scenario suite Format Editor. Display the scenario suite Format Editor and hold down Ctrl+S to save its content.

Step 4: Add the scenario suite format to the customization, repackage it, and redeploy both the SSP and Decision Center

Follow these steps to add the custom scenario suite to the customization and to repackage and redeploy both the SSP and Decision Center:

1. To add the scenario suite format to the customization, first display the Customization Editor by double-clicking the **Customization.sspc** file in the DVS Project (switch back from the Java Perspective to the Rule Perspective, if needed). See Figure 6-88 on page 193.

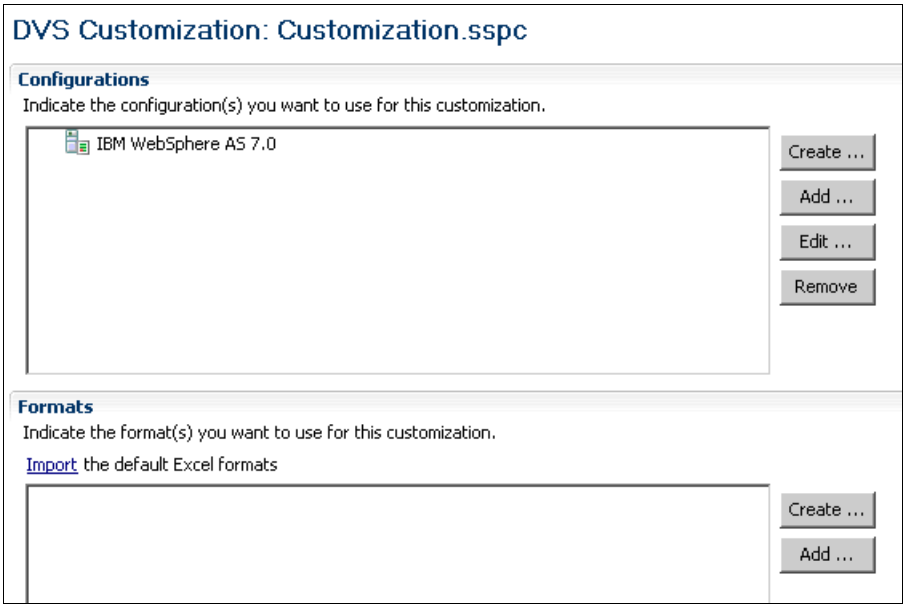


Figure 6-88 Customization Editor (Formats section)

2. In the Formats section of the editor, click **Add** to display the format selection window (Figure 6-89).

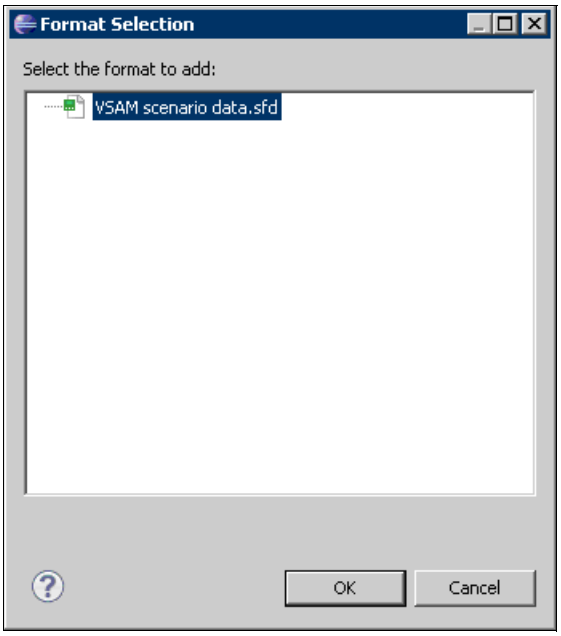


Figure 6-89 Format Selection window

3. Select the **VSAM scenario data.sfd** format and then click **OK**. The format selection window is closed, and the format is added to the customization (Figure 6-90 on page 194).

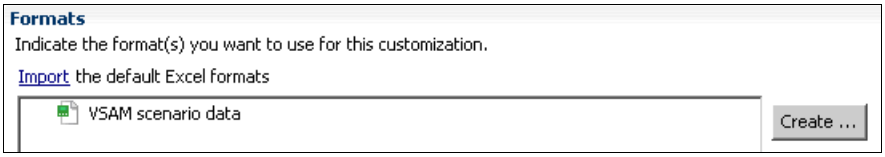


Figure 6-90 The VSAM scenario data scenario suite format is added to the customization

4. Save the content of the Customization Editor by pressing Ctrl+S. Click **Repackage** the .ear/.war files in the Actions section of the Customization Editor (Figure 6-91).

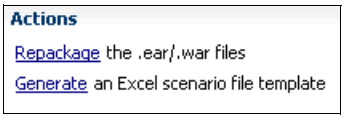


Figure 6-91 The Actions section of the Customization Editor

5. The Repackage SSP and Decision Center files for DVS window opens. Ensure that the **IBM WebSphere AS 7.0** configuration is selected. Also, ensure that both the **Repackage SSP .ear/.war file** and **Repackage Decision Center .ear/.war file** options are selected (Figure 6-92).

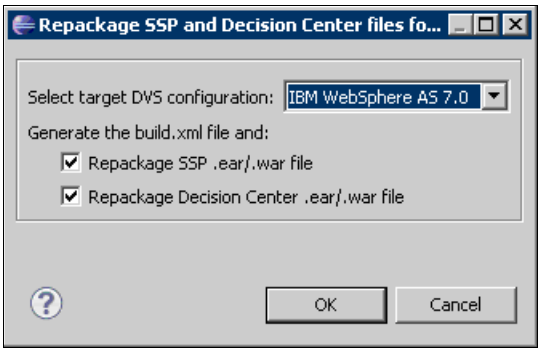


Figure 6-92 The Repackage SSP and Decision Center files for DVS window with selected options selected

6. Then, click **OK** and wait for the confirmation message that lists the paths of the two repackaged enterprise application archives (Figure 6-93).

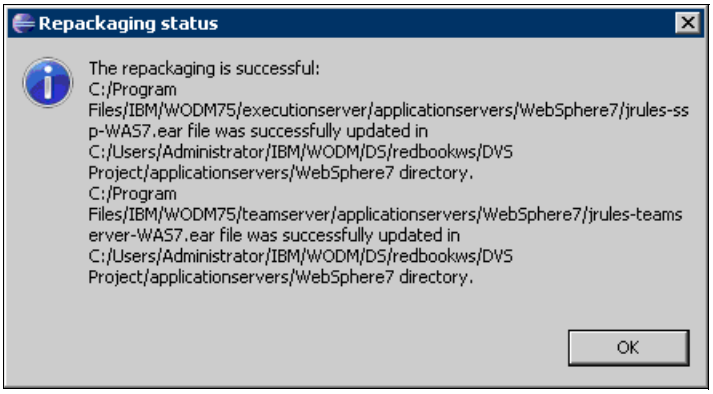


Figure 6-93 Confirmation of successful repackaging of both the SSP and Decision Center files

Next, we redeploy the repackaged SSP by following the instructions in the corresponding section of the *Decision Center Installation And Configuration Guide:*

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/index.jsp?topic=%2Fcom.ibm.dcenter.z.configuring%2FContent%2FBusiness_Rules%2F_pubskel%2FInfocenter_Primary%2Fps_DCz_Configuring3324.html

Step 5: Enable the new scenario suite format for the insurance-rules project in Decision Center

Follow these steps:

1. Sign in to Decision Center with an administrator profile (Figure 6-94).

Figure 6-94 Signing in to Decision Center

2. On the Home page of Decision Center, ensure that the project in use is the **insurance-rules** project (Figure 6-95 on page 196). After verifying that the insurance-rules project is the project in use, click the **Project** tab (not shown in Figure 6-95 on page 196) to display the Project menu.

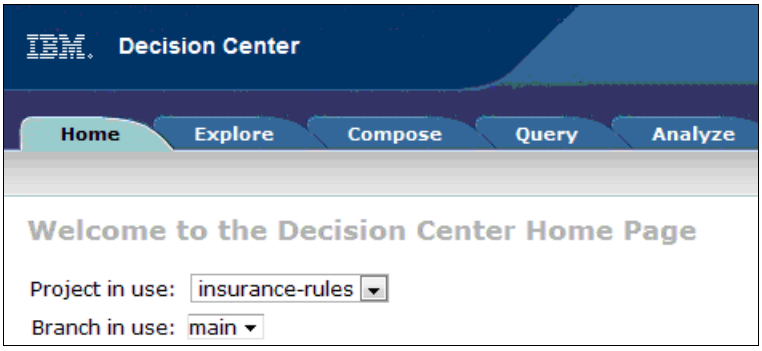


Figure 6-95 The Home page of Decision Center with the insurance-rules project in use

3. In the Project menu (Figure 6-96), click the **Edit Project Options** link.

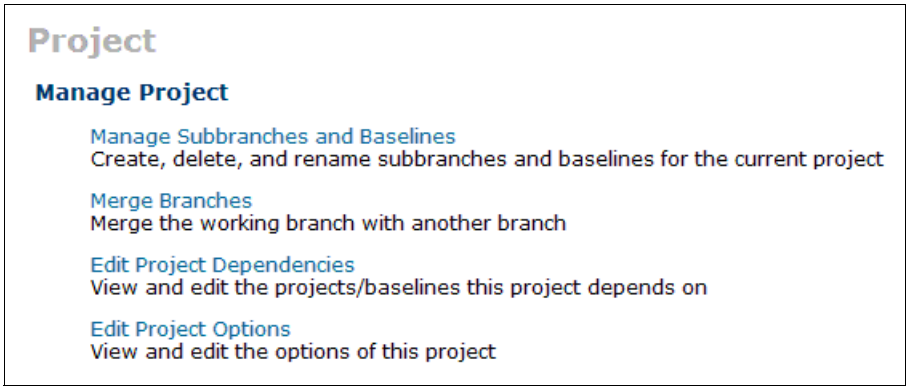


Figure 6-96 The Project menu in Decision Center

4. On the Edit Project Options page (Figure 6-97 on page 197), under the Test Suite / Simulation Options section, enable the **VSAM scenario data** format for simulations.

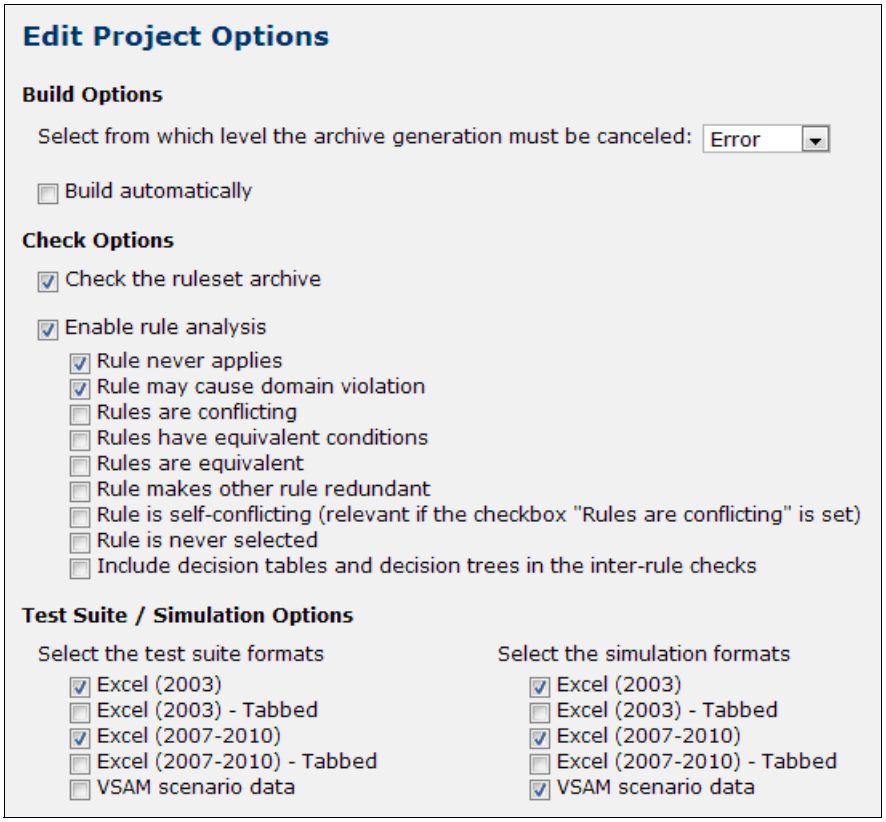


Figure 6-97 Enabling the VSAM scenario data format for simulations

5. Click **OK** (not shown in Figure 6-97) to enable the new simulation format.

Step 6: Create and run a simulation with the VSAM scenario data format in Decision Center

Follow these steps:

1. Click the **Compose** tab to display the Compose page in Decision Center (Figure 6-98).

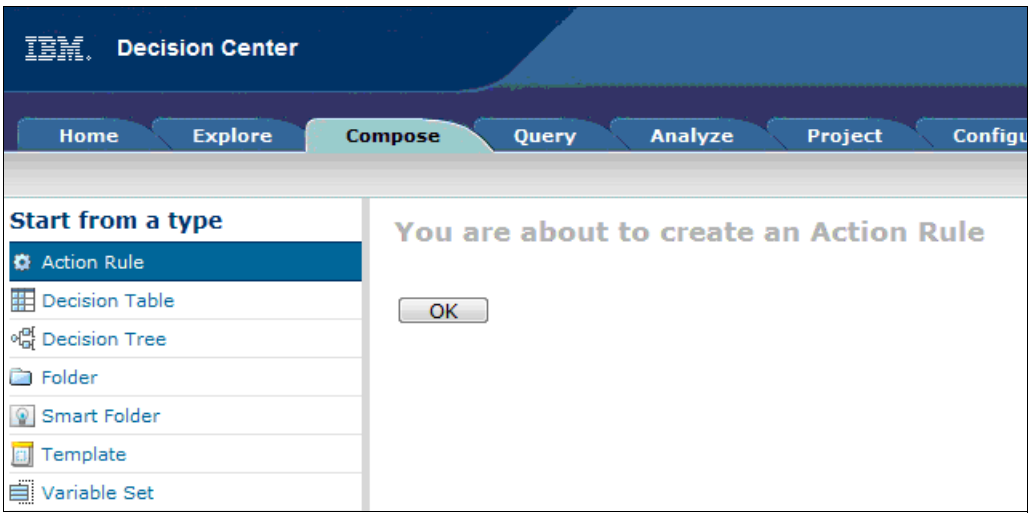


Figure 6-98 The Compose page in Decision Center

2. To change the type of artifact to create a simulation, click **Simulation** in the left menu (Figure 6-99).

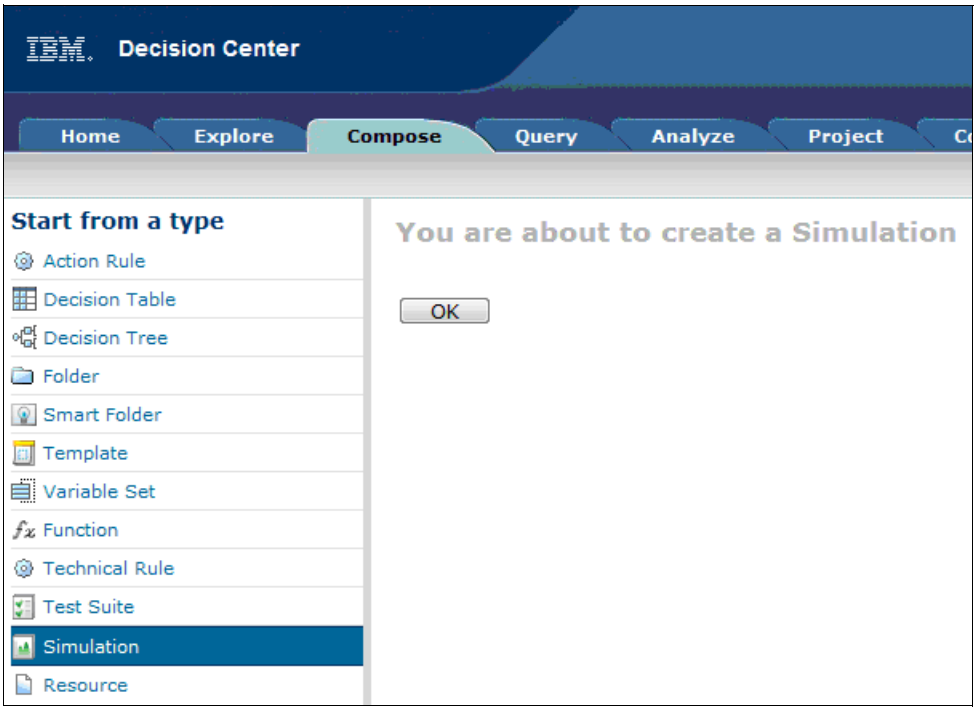


Figure 6-99 Select Simulation to create a simulation in Decision Center

3. The first page of the new simulation wizard appears (Figure 6-100). Keep the default name **New Simulation** for the simulation that we create. Click **Step 3: Scenarios** in the left menu.

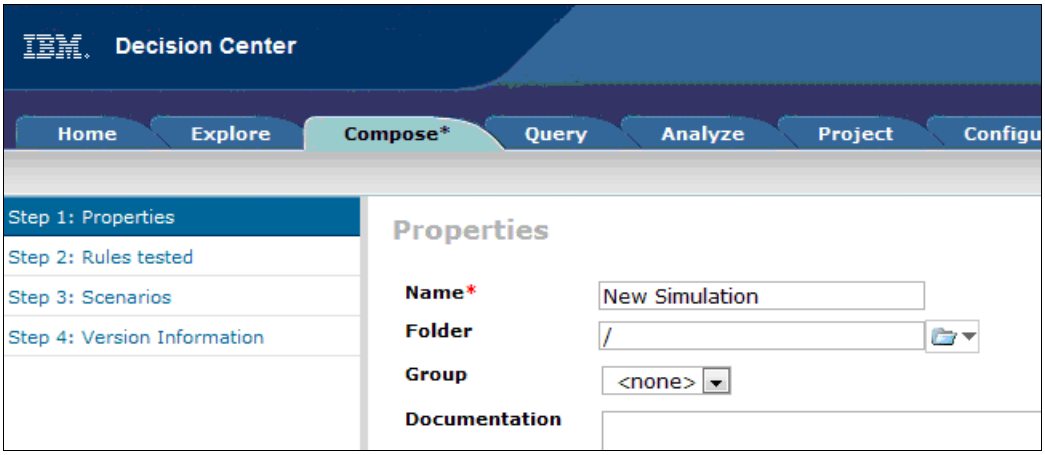


Figure 6-100 The first page of the new simulation wizard in Decision Center

4. On Figure 6-101 on page 199, select the **VSAM scenario data** format, and click **Next**.

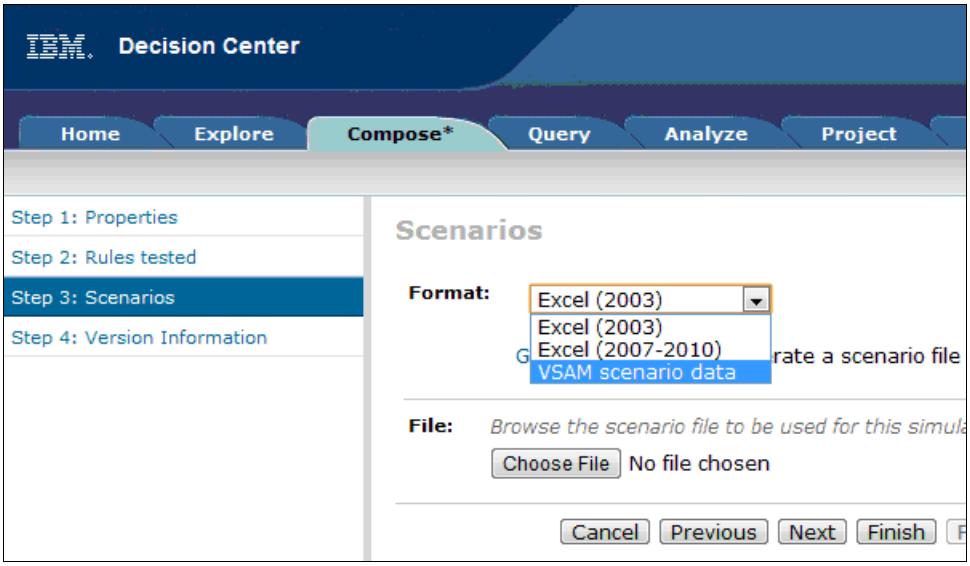


Figure 6-101 Selecting the VSAM scenario data format for a new simulation in Decision Center

5. The wizard page is updated to display the form that is defined by the custom scenario resource renderer that was defined for the format. For the name of the VSAM file, type `'SBRUNOT.ZL.DATA.VKSD0080'` to retrieve the scenario data (*do not forget to begin and end the file name with a single quote* (')). Type `1` for the index for the first scenario in the file. Type `2` for the index for the second scenario in the file (Figure 6-102). After you enter the scenario provider initialization data, click **Finish and Run** to save the new simulation and run it.

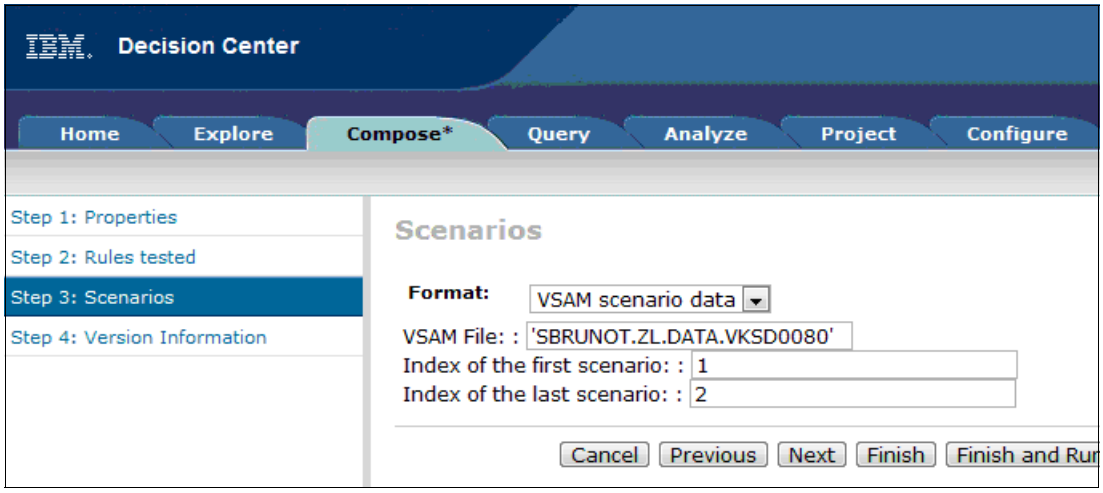


Figure 6-102 The scenario provider parameters form for the VSAM scenario data format

6. The runtime options for the simulation are displayed. On the Run New Simulation page that is shown in Figure 6-103, leave the default options selected and click **Run** to start the simulation.

Figure 6-103 The runtime options for a simulation in Decision Center

When the execution completes, the New Simulation execution report is displayed (Figure 6-104).

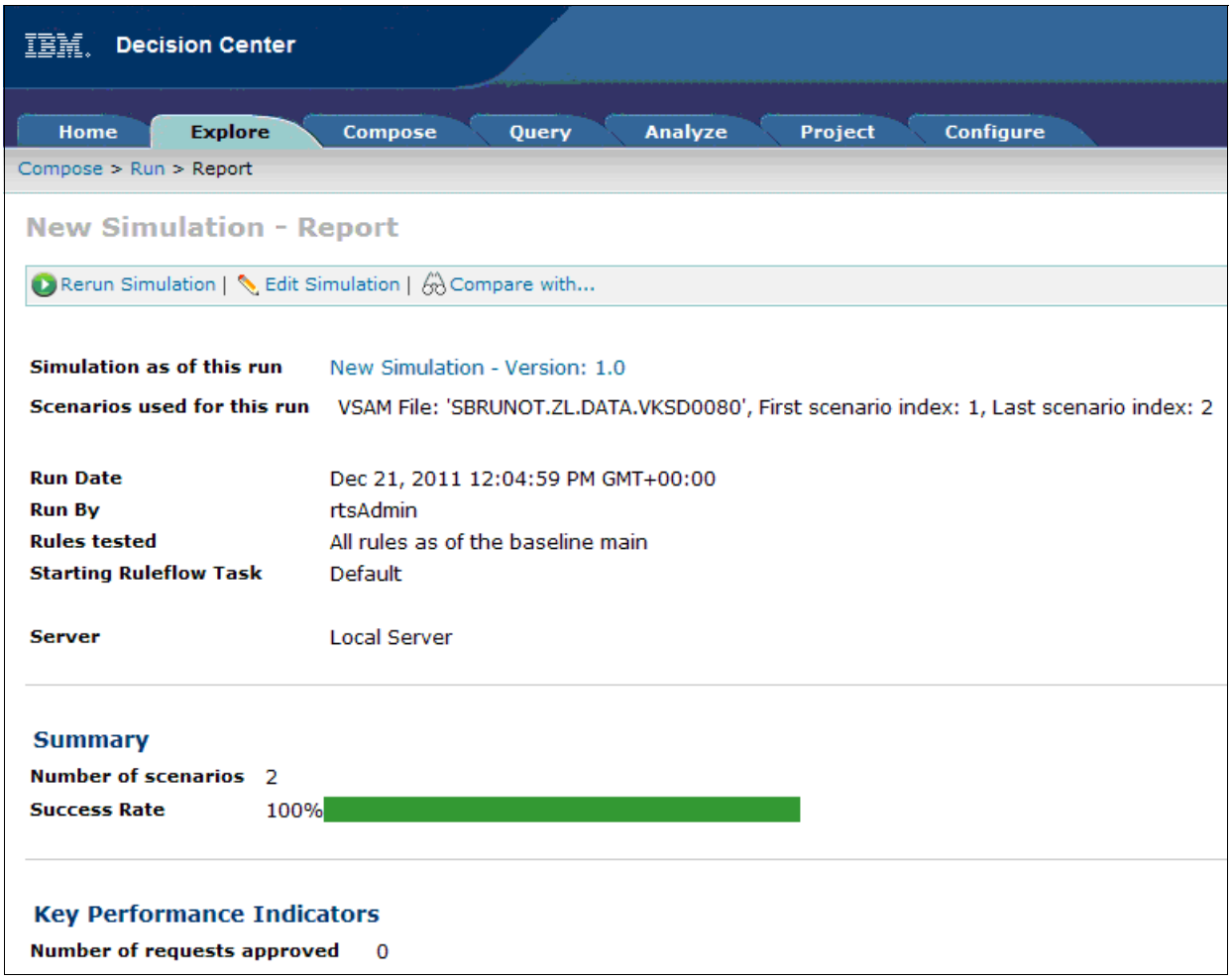


Figure 6-104 Simulation report in Decision Center

7

Advanced topics for decision authoring

In this chapter, we describe the authoring rules for deployment to z/OS. We do not explain general rule authoring. However, we describe in more detail the mapping between the COBOL data structures passed into the decision run time and the Java structures that are used to execute the rules. We also explain how you can customize that mapping.

This chapter also describes starting a new project for deployment from both a COBOL copybook and an existing Java-based rule project. We explain how you can extend the capabilities of the decision execution by adding custom methods into the business object model (BOM).

© Copyright IBM Corp. 2012. All rights reserved.

7.1 Designing the decision interface

The starting point for any successful decision project is to design the decision interface correctly. Often, we have an existing copybook that is used by the application that we are enabling to access WebSphere Operational Decision Management. The natural starting point seems to be to reuse this data structure as the interface to the decision. Although this approach might seem to be the easiest, there are benefits to be gained by giving the interface more consideration.

When designing the decision interface, consider why we are externalizing business rules into an external decision server. One of the main reasons is that the business decisions change on a shorter lifecycle than the applications that invoke them. Therefore, we want to isolate the application from changes to the business rules in the decision. However, we also want to isolate the business decision from maintenance changes that occur in the application, as well.

Changes in the application can alter the copybook that the application uses. Application changes result in the need to import the copybook again to update the BOM that is used in the rules. The changes potentially disrupt rules that are already authored.

Also, the data in the copybook might not be suitable for use with rule authoring. Application copybooks in COBOL are often a mix between a representation of the business data and its specific entries. For example, we often put COBOL FILLER statements in to align data, or we create fields to hold return codes or other diagnostic information. These fields have no business relevance to the decision and are confusing if included in the BOM.

It is also important to ensure that the data passed across to the business decision contains all the required information to successfully make the decision. When thinking about the data, consider information that might not be used in the business rules that are embedded in the application today. This information might be useful to develop a better business decision after it is externalized. It is far easier to pass more data across when designing the decision interface from the outset than to re-engineer the interface at a later date or add code into the decision to retrieve external data.

It is a preferred practice to design the interface to a business decision in the same way that you design a service interface. Look at the data that is used in the decision today. Look at the data that is easily available to the application. Look at the data that is required today and potentially in the future to maintain the decision after it is externalized. Then, create a copybook to hold that information specifically for the decision interface that can be versioned for that purpose. The additional cost of a few COBOL MOVE statements is outweighed by the flexibility that this approach provides in isolating changes in both the decision and the calling application.

7.2 Mapping from the COBOL copybook

This section contains the following sections:

- Structure of a COBOL-based rule project
- Supported COBOL data types
- Creating custom converters
- Mapping level-88 constructs into BOM domain types

7.2.1 Structure of a COBOL-based rule project

In Chapter 3, “Getting started with business rules” on page 27 when importing a COBOL copybook, we generated two Java projects: the Java Execution Module (XOM) project and the marshaller project.

XOM project

The XOM project contains a Java representation of the structure of the data in the COBOL copybook. Each level-01 item in the copybook, both group and elementary items, is mapped to a Java class. Each non-level-01 group item is also mapped to a class. Figure 7-1 is an example of this mapping.

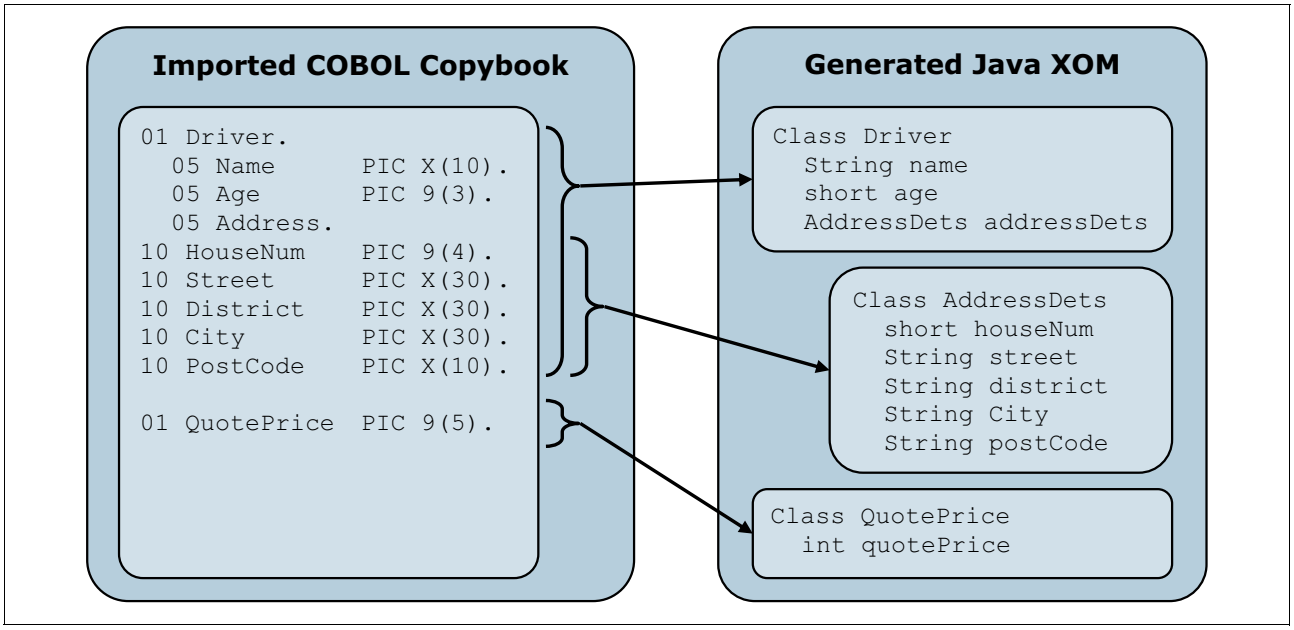


Figure 7-1 Generating a Java XOM from the COBOL copybook

Important: You can use the classes that are mapped from a level-01 group only as ruleset parameters that define the interface to the decision.

Marshaller project

The marshaller project contains the Java classes of the marshallers. During the processing of requests from a COBOL application, the zRule Execution Server for z/OS server first calls these marshallers to convert the COBOL data into Java XOM objects. After executing the ruleset, the zRule Execution Server for z/OS server calls the marshallers again to convert the Java objects back to COBOL data.

The marshaller classes are intended to be called by the zRule Execution Server for z/OS server only. You must not change the content of the generated code in the marshaller project, because any change to the code likely causes runtime errors.

The marshaller project is in the XOM path of the rule project, but you must not create a BOM entry for it.

Customizing the project and package names

For the generated XOM, the default project name is the COBOL XOM name with the suffix –xom. For the generated XOM, the default package of the generated class is xom. For the marshaller, the project name suffix is –marshaller. For the marshaller, the package name is marshaller. For example, if your XOM name is insurance, the XOM project name is insurance-xom, and the marshaller project name is insurance-marshaller.

We can change the project and package names when we import the copybook. In the first page of the wizard, we expand **Advanced**, as shown in Figure 7-2. At the bottom of the panel, in the Resource Configuration section, we can change the group default names.

It is important to ensure that you use a valid Eclipse project name or valid Java package name for each entry.

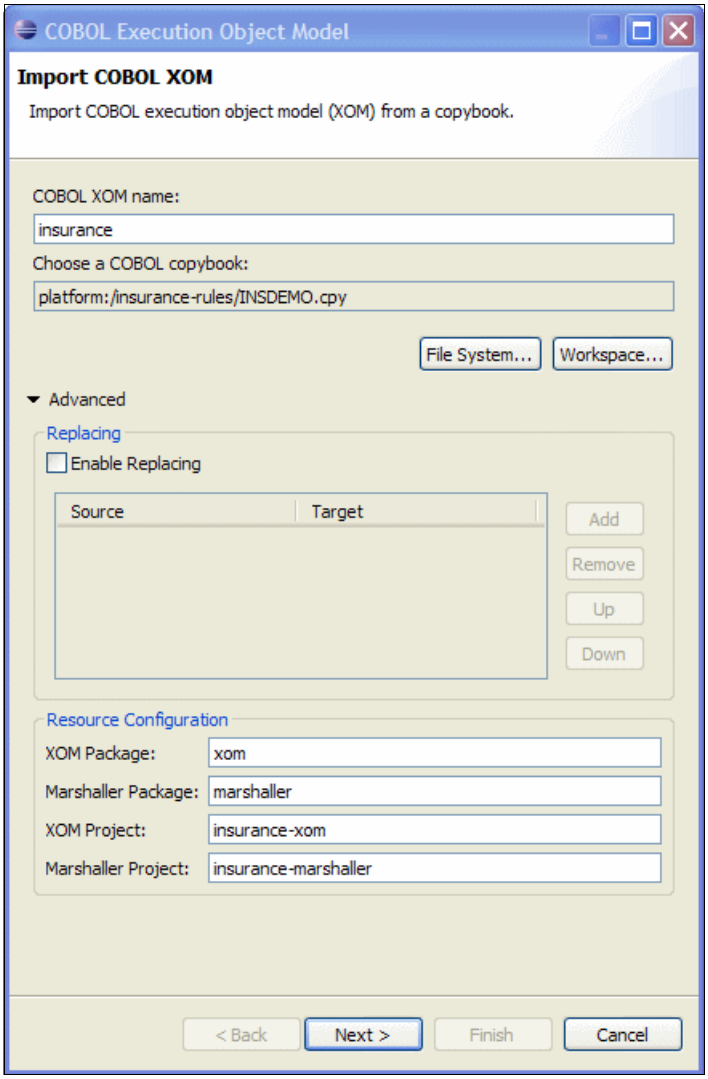


Figure 7-2 Configure project names and package names for XOM and marshaller

7.2.2 Supported COBOL data types

This section describes the supported and unsupported COBOL data types.

Basic mapping

Table 7-1 lists the supported COBOL-type to Java-type mappings.

Table 7-1 COBOL to Java mapping

COBOL type	COBOL usage and compile options	PICTURE string	Example	XOM Java type
AlphaNumeric	DISPLAY	Combination of A, X, and 9	PIC X(12)	String
Numeric	COMP-5 or BINARY, COMP, COMP-4 with TRUNC(BIN)	S9(1) through S9(4)	PIC S9 BINARY	short
		S9(5) through S9(9)	PIC S9(6) BINARY	int
		S9(10) through S9(18)	PIC S9(10) BINARY	long
		9(1) through 9(4)	PIC 9 BINARY	int
		9(5) through 9(9)	PIC 9(6) BINARY	long
		9(10) through 9(18)	PIC 9(10) BINARY	BigInteger
		9(10) through 9(18), with decimal (V or P)	PIC S999V9 BINARY	BigDecimal
	DISPLAY, COMP-3, PACKED-DECIMAL or BINARY, COMP, COMP-4 and not TRUNC(BIN)	S9(1) through S9(4) 9(1) through 9(4)		short
		S9(5) through S9(9) 9(5) through 9(9)		int
		S9(10) through S9(18) 9(10) through 9(18)		long
		S9(10) through S9(18) and 9(10) through 9(18), with decimal (V or P)		BigDecimal
	DISPLAY, COMP-3, PACKED-DECIMAL and ARITH(extend)	S9(19) through S9(31) 9(19) through 9(31)		BigInteger
	DISPLAY, COMP-3, PACKED-DECIMAL and ARITH(extend)	S9(19) through S9(31) 9(19) through 9(31) Decimal (V or P)		BigDecimal
DBCS	DBCS	G, B, or N with DISPLAY-1	PIC G(10)	String
InternalFloat	COMP-1			float
	COMP-2			double
Level 88				methods

COBOL type	COBOL usage and compile options	PICTURE string	Example	XOM Java type
National	NATIONAL	PIC N(8)		String
		PIC 999V9 SIGN LEADING SEPARATE, SIGN TRAILING SEPARATE		BigDecimal

Unsupported types

Table 7-2 lists the unsupported types.

Table 7-2 Unsupported COBOL types

COBOL type	COBOL usage and compile options	PICTURE string	Example
Alphabetic	DISPLAY	A	PIC A(20).
AlphaNumericEdited	DISPLAY	A X 9 B 0 /	PIC XBX.
NumericEdited	DISPLAY	B P V Z 9 0 / , . + - CR DB * cs	PIC 9B9
ExternalFloat	DISPLAY	+9 -9 0 . V E 9.	PIC +99V9E99.
NationalExternalFloat	NATIONAL	PIC +9.9E+99	PIC NBN
NationalEdited	NATIONAL	PIC NBN PIC $9.9	

These types are not supported, because there are no suitable Java types to which to map. We advise that you do not include these types in the import copybook. If the copybook structure cannot be changed, for example, for compatibility with existing applications, consider this work-around. In the copybook for importing, change the data item to a corresponding ordinary alphanumeric or national type of the same length. Then, these data items are mapped to Java strings in the generated XOM class.

There are also two unsupported Occurs Depending On (ODO) table situations:

- ODO table within a fixed-length table
- ODO table sharing the ODO object

Consider using a fixed-length table to work around these limitations.

Converter

You can use type converters to change the Java type to which a COBOL data item is mapped. There are two built-in converters: String to boolean converter and String to Date converter. You can also implement custom converters or set an XOM field to a custom-defined domain class.

7.2.3 Creating custom converters

In Chapter 3, “Getting started with business rules” on page 27, you learned how to use the built-in type converters to map a COBOL data item to Java boolean or Date type. There are cases when the built-in converter cannot meet your need. Then, you can write a custom converter.

Consider this case. In a COBOL program, instead of using only `T` to indicate true, the program also accepts `t`, `Y`, and `y` as true values. But the built-in boolean converter can accept only one character as the true value. So, in this case, we write a custom converter.

A *custom converter* is a normal Java class. The class must be annotated with the TypeConverter annotation. In the Converter dialog, users can select those classes with only the TypeConverter annotation, along with the built-in converters.

Then, we need implement an init method. This method is called by the run time to initialize the converter with user-defined properties. There is also an optional TypeConverterProperties annotation with which we can define the list of expected property keys. In this example, we define two keys:

true-values	A comma-separated list of characters that indicate true
false-value	The default value that indicates false

In the init method, we retrieve the values of these properties. Then, we save the list of true values to the trueValues field and set the falseValue field to the false-value character (Example 7-1).

Example 7-1 Custom converter code (Part 1 of 2)

```
@TypeConverter
@TypeConverterProperties({ "true-values", "false-value" })
public class MyBooleanConverter {
   private List<String> trueValues;
   private String falseValue;

   public void init(Map<String, String> props) {
      String trueStr = props.get("true-values");
      trueValues = Arrays.asList(trueStr.split(","));
      falseValue = props.get("false-value");
   }
```

Now, we can define two conversion methods:

```
public <TargetType> convertToTarget(<SourceType> value)
public <SourceType> convertToSource(<TargetType> value) {
```

The <*SourceType*> is the default Java type that directly mapped from COBOL data; the <*TargetType*> is the type that we want to use in the generated XOM. The convertToTarget is called during unmarshalling, and the convertToSource is called during marshalling.

In Example 7-2, if the value that comes from COBOL is within the trueValues, the result is true. Any other values convert to false. During marshalling, if the Java value is true, the first of the possible true values is used, which is `T` in this case. If the Boolean value is false, the false value `F` is used.

Example 7-2 Custom converter code (Part 2 of 2)

```
   public synchronized boolean convertToTarget(String value) {
      return trueValues.contains(value);
   }
   public synchronized String convertToSource(boolean value) {
      return value ? trueValues.get(0) : falseValue;
   }
}
```

After the custom converter is implemented, we must add the Java project to the XOM path of the rule project. Then, when adding a COBOL XOM, in the Converter dialog, we can choose the converter that we defined and set the property values (Figure 7-3).

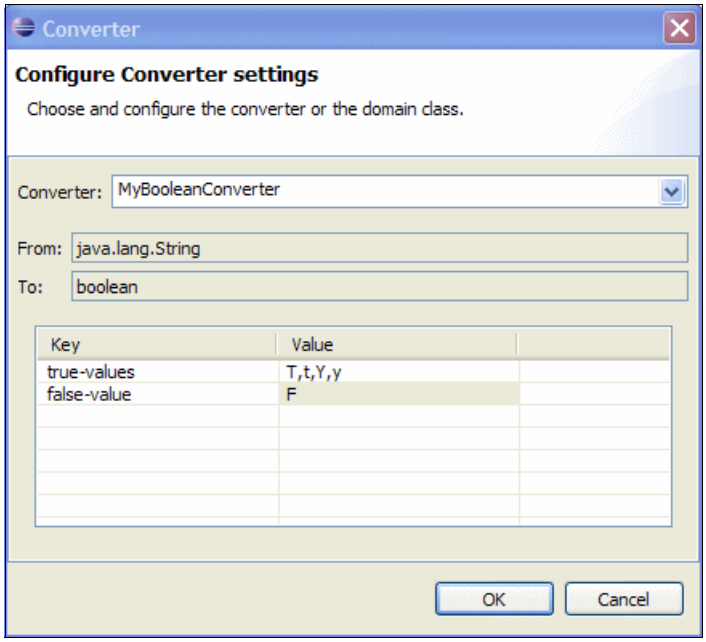


Figure 7-3 Converter dialog with custom converter

In the generated marshaller code, the user-provided properties in this dialog are sent as parameters in the call of the init method. So, the converter is initialized before any conversion operation.

7.2.4 Mapping level-88 constructs into BOM domain types

A *domain* can restrict the possible values that a type element in BOM can accept. During rule authoring, the editor suggests values according to the enumerated domains. A semantic check is also performed to check that the business rule does not use a value outside the defined domain.

Now, we look at the following copybook (Example 7-3), which we used in the previous example.

Example 7-3 COBOL copybook

```
05  VEHICLE.
 10  VEC-ID                    PIC  X(15).
 10  MAKE                      PIC  X(20).
 10  MODEL                     PIC  X(20).
 10  VEC-VALUE                 USAGE COMP-1.
 10  VEC-TYPE                  PIC  X(2).
     88 SUV     VALUE 'SU'.
     88 SEDAN   VALUE 'SD'.
     88 PICKUP  VALUE 'PU'.
```

The vehicle type VEC-TYPE data item has three level-88 items, which define the valid values for this item: `SU`, `SD`, and `PU`. When importing this copybook into Rule Designer, the level-88 items are mapped to methods. These methods are helpful to check the value of the field or to

assign the correct value to the field, but they cannot prevent the field from being assigned incorrect values. Ideally, a user might use valid values only with the vehicle type, or the user can use actual vehicle types instead of codes to represent the vehicle types. We must define a domain to meet this requirement. In the COBOL XOM import wizard, we define a domain converter to map the vehicle type to a Java domain type.

First, we need implement a Java class as the XOM for the domain definition. Example 7-4 is the sample code.

Example 7-4 Java class for domain vehicle type

```
package redbook;
public class VehicleType {
  public final static VehicleType SUV = new VehicleType("SU");
  public final static VehicleType SEDAN = new VehicleType("SD");
  public final static VehicleType PICKUP = new VehicleType("PU");
  private String code;
  public VehicleType(String code) {
    this.code = code;
  }
  public String getValue() {
    return code;
  }
}
```

In this class, we define a constructor with a string parameter. This constructor accepts the vehicle type codes and creates a VehicleType object. We also must implement a getValue method to retrieve the string code from the VehicleType object.

In Rule Designer, we add the project to the Java XOM path of the rule project. Then, when we import a COBOL copybook, we define a converter for the vehicle-type item (Figure 7-4).

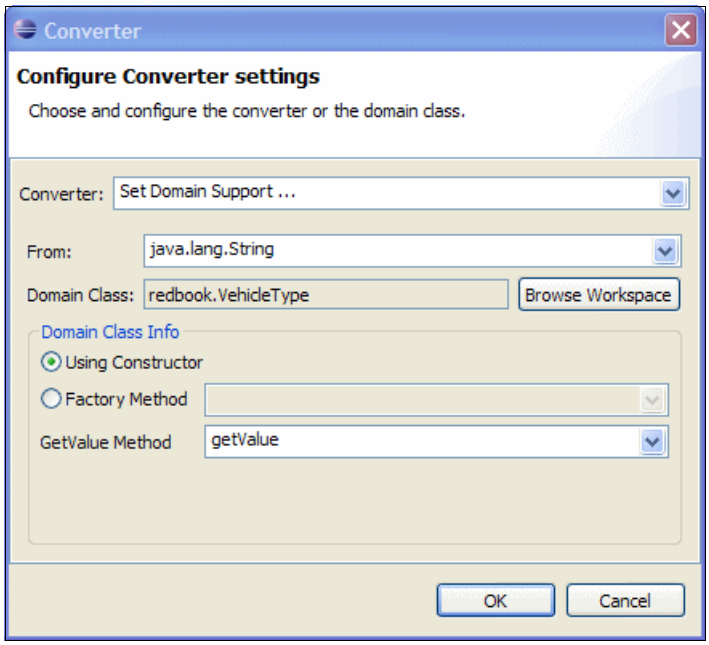


Figure 7-4 Converter dialog to set up the domain type

In the dialog that is shown in Figure 7-4 on page 209, when we select **Set Domain Support…** for the converter, we can choose the From type. The From type is the Java type when the COBOL data is first unmarshalled and before the converter is applied. We provide the domain class name that we want to use. Then, we provide detailed information for the domain class. We select **Using Constructor** to convert the string codes to the domain object. And, we select the **getValue** method to convert the domain object to string.

We must create a BOM entry for the domain class first. Then, we create a BOM entry for the COBOL XOM. With the correct verbalization, we can use the domain in rule authoring (Figure 7-5).

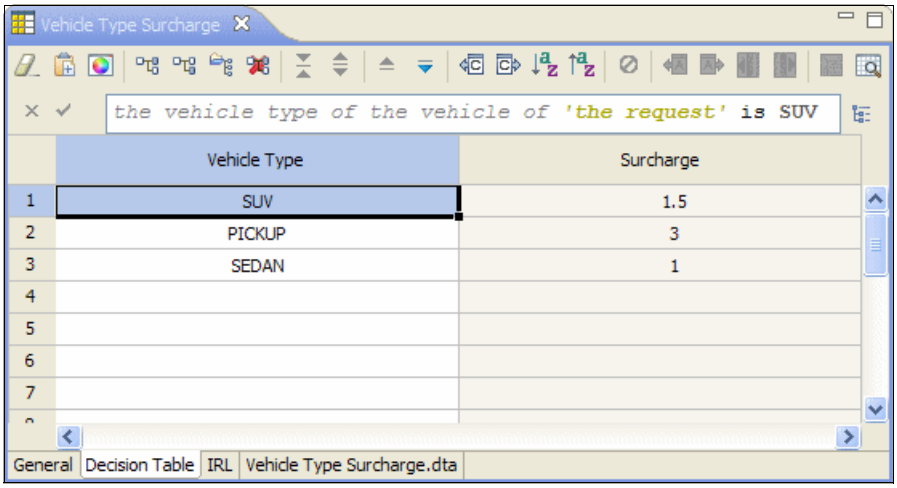


Figure 7-5 A sample decision table using the COBOL domain

In the converter dialog, we can also use a static factory method instead of a constructor to define a domain converter. Example 7-5 is a sample implementation of a static factory method.

Example 7-5 Static factory method in the domain class VehicleType

```
public static VehicleType getVehicleType(String code) {
    if (SUV.code.equals(code))
      return SUV;
    else if (SEDAN.code.equals(code))
      return SEDAN;
    else if (PICKUP.code.equals(code))
   return PICKUP;
    else
   return null;
  }
```

7.3 Starting from an existing Java-based BOM project

If rule projects are currently in production on distributed systems, and you want to migrate the rule application to the mainframe, you can enable the BOM for COBOL application and generate a COBOL copybook.

Only a BOM that originated from a Java XOM is supported. A BOM that originated from an XML-based dynamic XOM is not supported.

7.3.1 Mapping Java data structures to COBOL

This section explains mapping Java data structures to COBOL.

Aggregation data structure

When mapping Java BOMs to COBOL, only BOM classes with aggregation relationships are supported. If there are object references, a simple hierarchical data structure is supported. Complex object graphs are not supported. For example, the Insurance class has a field called Vehicle of type Vehicle. The vehicle information is part of the insurance data, and it is a simple hierarchical data structure. This example is supported.

But, if the Vehicle class references Insurance either directly or indirectly through other classes, the data structure is not hierarchical. It contains loops. In this case, the BOM to COBOL mapping is not supported. In addition, class inheritance is not supported. The BOM must not include the following usage:

- Inheritance
- Loop reference, including self-reference
- Static attribute

Also, ensure that the BOM classes follow Java Bean naming guidelines, such as well-formed getters and setters; otherwise, generated marshaller classes contain incorrect code.

General mapping rule

A BOM class is mapped to a COBOL group. Fields of the basic Java type are mapped to child elementary data items. And, fields of the class type are mapped as subgroups (Example 7-6).

Example 7-6 A BOM with two classes

```
package xom;
public class Request {
    public xom.Driver primaryDriver;
    public xom.Driver secodaryDriver;
}
public class Driver {
public short age;
public string name;
}
```

When this BOM is mapped to a COBOL copybook, the primaryDriver and secodaryDriver fields are generated as two groups with the same structure (Example 7-7).

Example 7-7 Copybook with two similar groups

```
01 request.
           02 primaryDriver.
               03 age pic S9(5).
               03 name pic X(20) value SPACE.
           02 secodaryDriver.
               03 age pic S9(5).
               03 name pic X(20) value SPACE.
```

An array is mapped to a table, and a collection is mapped to a size data item and a table (Example 7-8 on page 212).

Example 7-8 A BOM with an array and list

```
package xom;
public class Request {
    public xom.Driver[] drivers;
    public java.util.List vehicles domain 0,* class xom.Vehicle;
}
public class Driver {
public short age;
public string name;
}
public class Vehicle {
    public string vehicleId;
    public double vehicleValue;
}
```

The drivers field is an array, so the COBOL data item is a fixed-length table. The vehicles field is a list of Vehicle objects. So, in the generated copybook, vehicles-Num is used as the actual size of the table vehicles (Example 7-9).

Example 7-9 Copybook for array and list sample

```
01 request.
           02 drivers Occurs 10 Times.
               03 age pic S9(5).
               03 name pic X(20) value SPACE.
           02 vehicles-Num pic 9(9).
           02 vehicles Occurs 10 Times.
               03 vehicleId pic X(20) value SPACE.
               03 vehicleValue usage COMP-2.
```

Mapping the basic Java type

Table 7-3 lists the Java to COBOL mapping.

Table 7-3 Java to COBOL mapping

Java type	Default COBOL mapping	Configuration
byte	S9(3)	Sign and length USAGE BINARY, PACKED-DECIMAL, COMP-5
short	S9(5)	
int	S9(10)	
long	S9(18)	
java.math.BigInteger	S9(18)	
float	COMP-1	Sign and length USAGE BINARY, PACKED-DECIMAL, COMP-5, and COMP-1
double	COMP-2	
java.math.BigDecimal	S9(9)V9(8)	
java.lang.String	X(20)	X/N, length
java.util.Date	9(8) [*yyyyMMdd*]	9/X, date format
boolean		

You can change the default Java to COBOL mapping in the Default COBOL Type Setting tab on the COBOL Code Generation property page.

The following classes are unsupported Java types:

- Any Java classes, except the classes that are listed in Table 7-3 on page 212
- The java.lang.Object class
- Classes that are defined in another BOM entry

7.4 Extending the capability of the rule execution with BOM methods

The Business Action Language (BAL) that is used to define rules is flexible and extensible. Generally, business rules are written from a vocabulary that is based on the structure of the data that is passed in, for example:

```
If the age of the borrower is less than 18
then ……
```

It is also possible to verbalize methods to be invoked from rules, as well. These methods can come from either methods from the imported Java XOM classes or defined directly in the BOM as virtual methods, for example:

```
public void rejectTheLoan()
{
    this.approved = false;
}
```

The method in this example can be verbalized as reject the loan and then used for rule authoring:

```
If the age of the borrower is less than 18
then reject the loan ;
```

Here, the method ‘reject the loan’ can be used in place of the BAL statement:

```
make it false that the loan is approved ;
```

This approach greatly simplifies the rule authoring experience.

Although BOM methods are useful, ensure that the business decision can still be managed by the business and that the decision is still reusable across multiple platforms. In this section, we describe several of the preferred practices for BOM methods and show you an example of using them.

7.4.1 Preferred practices for using virtual methods

To the IT-focused decision developer, the BOM methods might appear to be an attractive way to augment the capabilities of a business decision. However, you can negate the value of externalizing a business decision into a business rule engine if you do not use the BOM methods correctly. We describe a few of the preferred practices to help you avoid misusing the BOM methods.

Do not bury business logic in the business decision

When we realize that we can access custom Java code from within a business decision call, it can be tempting to add business logic to the decision. Adding business logic to the decision is

generally a bad idea. This statement might seem counterintuitive, because often the terms business logic and business rules are used interchangeably. When looking at the business decisions, we must consider them only the *rules* part of the business application. Figure 7-6 shows a simplified representation of a business application.

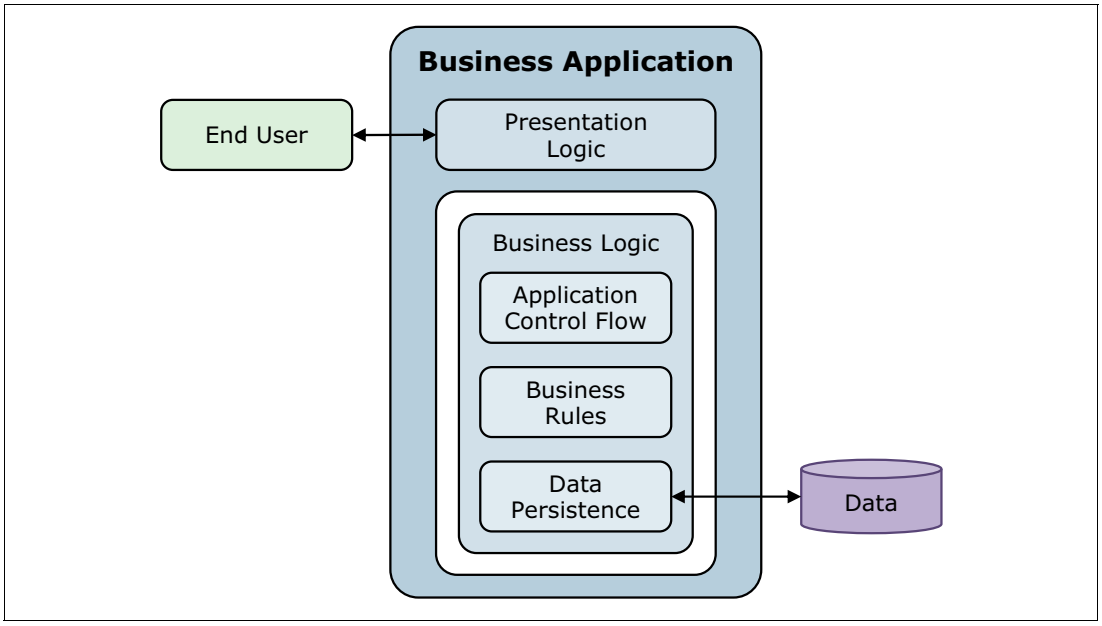


Figure 7-6 The structure of a business application

In Figure 7-6, we see that a business application is normally made up of the following elements:

- Presentation logic handling the user interaction
- Application control flow handling the flow of the logic through the application
- Business rules that are the implementation of the business behavior
- Data persistence layer that handles interaction with data sources

It is important to point out that the business rules do not interact with anything outside of the application. If we want to be able to hot-deploy new versions of decisions, we must be certain that changes to the business rules cannot break the application. If the business rules call out to external data sources, any changes must be tested within the full application, forcing a full regression test. If the changes to the business rules change only the business behavior within the application, we can test the business rules in isolation to the full system. We only test to ensure that the rules implement the business requirement correctly. We ensure that the rules do not cause an issue for the application, for example, by forcing a divide by zero error. This level of testing is only appropriate if the business rules are encapsulated within the application.

Do not add platform-specific logic if sharing rules

In most cases, Java is platform-independent. However, it is still possible to code Java methods that only run in certain environments. For example, the JzOS Java libraries, which are part of the base Java Runtime Environment (JRE) 6.0 for z/OS, provide a collection of methods that are useful when coding Java on the mainframe. They contain methods for accessing z/OS resources and formatting data, plus other useful features. After you code a BOM method that uses the JzOS libraries, this business decision cannot be reused on a distributed platform. The only way to share is to create either two versions of the Java XOM or two separate rulesets, each containing a separate implementation of the BOM method,

depending on where the method is coded. Both these options lead to greater complexity in the rule management that is required to keep consistent decisioning across the platforms.

Use BOM methods sparingly

One of the key values of externalizing your business decisions as business rules is the ability to author them in natural language, making them accessible to the business team. If we use too many BOM methods, the result is recoding the business decision from the application into BOM methods in the business rules rather than as BAL rules that are accessible. In the extreme case, it is possible to code so much of the logic in BOM methods that the externalized business decision is no more accessible to the business team than the original business application from which the decision was extracted. The key is to use BOM methods sparingly where they add value to the rule authoring by simplifying the language and the logic required to author the decision.

BOM methods are useful in the following examples:

- Coding a formula that does not change but is used repeatedly within the decision

  One example is calculating the after-tax income of a customer where the tax amount is available as a variable to the BOM method.

- Simplifying the language that is required to perform a business operation to abstract from the data model

  An example of this BOM method is where the business user can:

  `reject the loan rather than having to know the relevant part of the data model to alter to create this behavior`

- Handling more complex data structures within the data model

  The XOM model can contain more complex data structures, such as ordered lists. You can use a simple BOM method, such as addMessage(), to isolate the business user from this complex data structure.

7.4.2 Calling out from a ruleset to a VSAM file to augment data

Occasionally, you are required to call out from a ruleset to augment the data that is required to make the decision. Only call out from a ruleset to augment the data in these circumstances:

- The data is not easily accessible to the application program.
- The decision cannot be made without the data.
- The data is required only in exceptional circumstances by the decision.

The JzOS libraries that come with IBM Java Runtime Environment 6 for z/OS provide a set of classes that can be used to access a record from a VSAM file. The following code shows an example of using the JzOS library classes to read in a specified record from a VSAM file. You can use this code within a BOM method to augment the data that is available to a decision. To use the JzOS classes within a BOM method, you must first copy the JzOS library to your workstation and make it available to the rule project in which you want to author your rules.

The JzOS library is in your z/OS installation:

`<JAVA6_INSTALL_ROOT>/lib/ext/ibmjzos.jar`

To allow these classes to be available to your BOM authoring, you must add this jar file as an external jar to the Java Execution Model for your project. Use the project Properties panel, as shown in Figure 7-7 on page 216.

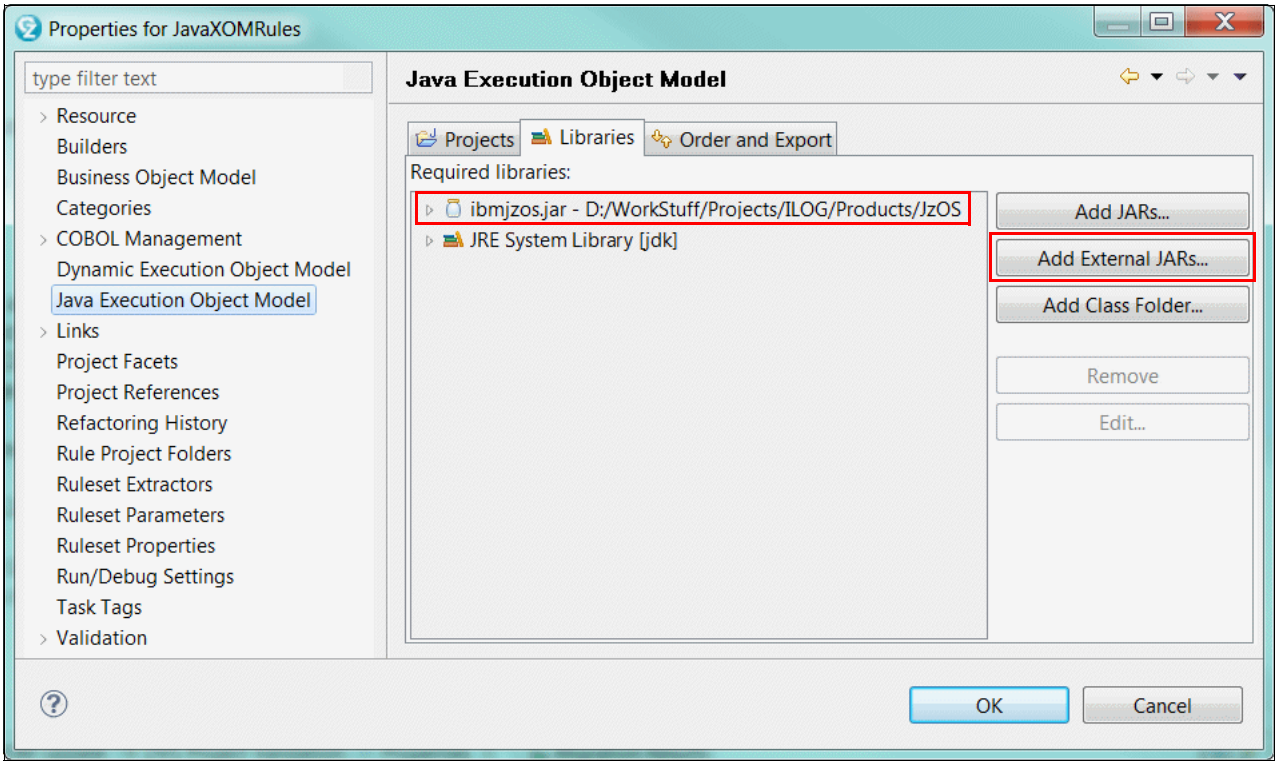


Figure 7-7 Adding the JzOS jar to the project class path

The class ReadKsdsVsam has a constructor and one method on it (Example 7-10). This class is designed to read a single row from a key-sequenced data set (KSDS) format VSAM file, based on a supplied record key, and to return the corresponding row as a byte array. By knowing the format of the record structure, the required data can then be read out from the byte array and copied into local rule variables. The byte array can be read simply by using Java substringing. Or if the record is more complex, the byte array can be read by using the Java record framework tooling. The Java record framework tooling is available in IBM Rational® Application Developer or IBM Rational Developer for System z.

Example 7-10 ReadKsdsVsam.java

```
import java.io.UnsupportedEncodingException;
import com.ibm.jzos.ZFile;
import com.ibm.jzos.ZFileException;
public class ReadKsdsVsam
{
   private ZFile zFile;
   private String filename;
   private String options;
   private int lrecl;
   private int keyLen;
   private byte[] keyBytes;
public ReadKsdsVsam(String filenameInput, int lreclInput, int keyLenInput) throws
ReadKsdsVsamException
{
   // Set the options to open the file as VSAM type file, read only
   options = "rb,type=record";
   this.filename = filenameInput;
   this.lrecl = lreclInput;
```

```
   this.keyLen = keyLenInput;
   // Format the file name
   if(!filename.startsWith("//"))
   filename = "//" + filename;
   try { // Check the file exists
      if(!ZFile.exists(filename)) {
         throw new ReadKsdsVsamException("File "+filename+" does not exist");
      }
         // Open the file
         zFile = new ZFile(filename, options);
      }
      catch (ZFileException zfe)
      {
         throw new ReadKsdsVsamException(zfe);
      }
   }
}
public byte[] readRecord(String key) throws ReadKsdsVsamException
{
   byte[] record = new byte[lrecl];
try {
      keyBytes = key.getBytes(ZFile.DEFAULT_EBCDIC_CODE_PAGE);
      boolean located = zFile.locate(keyBytes, 0, keyLen, ZFile.LOCATE_KEY_EQ);
      if (!located) throw new ReadKsdsVsamException("Record: "+key+" cannot be
found");
      zFile.read(record);
   }
   catch (ZFileException zfe) {
      throw new ReadKsdsVsamException(zfe);
   }
      catch (UnsupportedEncodingException uee) {
      throw new ReadKsdsVsamException(uee);
   }
   return record;
}
public void closeFile() throws ReadKsdsVsamException
{
   try {
      zFile.close();
   }
      catch (ZFileException zfe) {
      throw new ReadKsdsVsamException(zfe);
   }
}
}
```

In Example 7-10 on page 216, the constructor `public ReadKsdsVsam(String filenameInput, int lreclInput, int keyLenInput) throws ReadKsdsVsamException` takes the following arguments:

filenameInput	The fully qualified name of the file to read
lreclInput	The length of the record
keyLenInput	The length of the record key

The constructor checks to see whether the supplied file exists and then opens the file for reading. If any of these operations fail, it throws a `ReadKsdsVsamException` with the reason for

the failure. This method can be verbalized and called in the initial actions section of the ruleflow.

To read a record from the VSAM file, you then use the following method:

```
public byte[] readRecord(String key) throws ReadKsdsVsamException
```

This method takes in the key of the record to read as a String and returns the contents of the record (including the key) as a byte[] array in the code page in which it was read.

Finally, the close()method is called to close the file. This method can be verbalized and called in the final actions section of the ruleset ruleflow.

7.5 Considerations for sharing rules between z/OS and distributed applications

One advantage of externalizing your business decisions is that you can identify decisions that are duplicated across multiple applications. The next logical step is to remove the duplication and manage the duplicated decisions as one decision to ensure consistency across the solution. Situations can occur where a decision is deployed for both a z/OS-based application and a distributed application. This type of deployment is possible, although certain considerations exist.

7.5.1 Sharing a COBOL-based project with Java applications

When you start from a COBOL copybook as your definition for the data that is passed into the decision (the XOM), the tooling initially generates a Java representation of the COBOL data structure. These Java objects are used at run time by the decision server for evaluating the business rules. After you import the copybook and develop your rules, the following artifacts are left:

- The copybook
- A Java project that represents the data from the copybook
- A Java marshaller project
- One or more rulesets that define the rules in the decision

All these artifacts are required to deploy the rules to the zRule Execution Server for z/OS environment, as shown in Figure 7-8 on page 219.

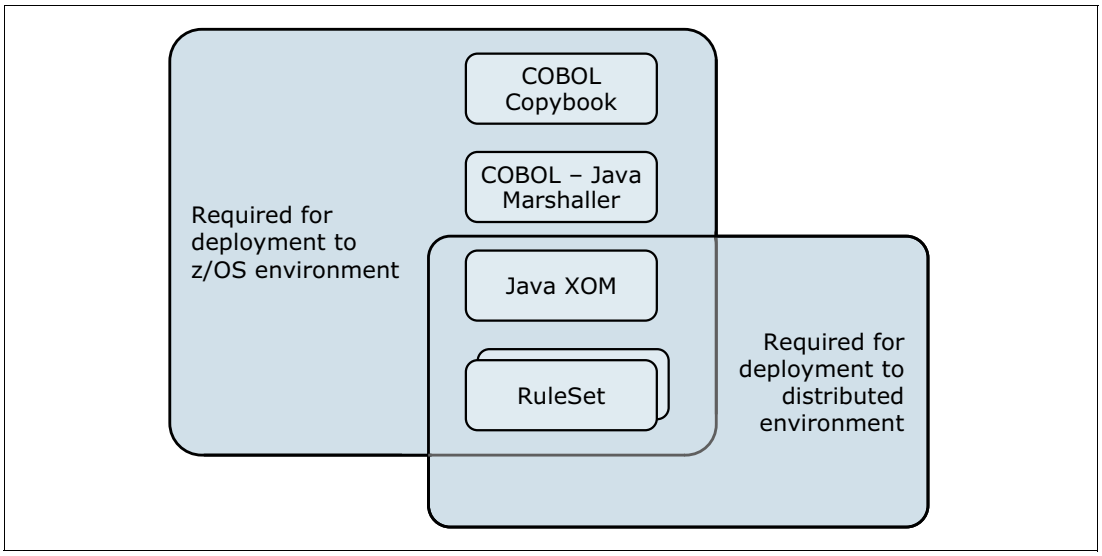


Figure 7-8 Artifact deployment

If you want to reuse this decision in a distributed environment, deploy the Java project that was created from the COBOL and the rulesets together to a distributed version of WebSphere Operational Decision Management. In this case, you use the standard Java APIs to access the decision. The client passes the data into and out of the decision using the Java objects that are generated from the COBOL copybook.

You must not edit or change the generated Java classes in any way. Any change to the COBOL copybook requires the regeneration of the Java XOM. Any changes that you made are lost. In this example, consider the COBOL copybook as the master copy of the data model. Any required changes must be made to the COBOL copybook, and all other artifacts regenerated.

7.5.2 Sharing a Java BOM-based project with COBOL applications on z/OS

In 7.3, “Starting from an existing Java-based BOM project” on page 210, we describe the process of enabling a Java BOM-based project for use with COBOL applications on z/OS. When planning to enable a Java-based BOM project, it is important to consider the restrictions on the Java types that can be supported in this process. It is also important to note that new artifacts are created in this process. The most important artifact to the run time is the Java project that contains the code to marshal between the COBOL data structures and the Java used at run time. This artifact must be deployed to the zRule Execution Server for z/OS run time with the Java XOM and ruleset projects.

When any changes are made to the Java XOM or to the generated BOM, you must rerun the process to update the COBOL artifacts to synchronize them with the Java changes. In this case, consider the Java XOM as the master data model. You must not change the COBOL copybook after it is generated.

When using an existing Java project as a starting point to deploy to z/OS, ensure that no platform-specific code is in the Java XOM. As described in 7.4, “Extending the capability of the rule execution with BOM methods” on page 213, rules can invoke methods that exist on classes in the Java XOM. Ensure that if any methods are used in the rules, the methods do not invoke any platform-specific code. Java code is independent of any platform. However, Java code can become specific to a platform if it tries to access a data source that exists only in the solution on particular servers. For example, the solution might use a customer record

database that is hosted locally to the decision execution on IBM AIX® server1 but is not accessible to z/OS server2 due to the firewall configuration.

7.6 Coding the COBOL client application

To access the zRule Execution Server, the COBOL application must use the supplied client API. This API consists of the following API calls:

HBRCONN To connect to the server
HBRRULE To execute a decision
HBRDISC To disconnect from the server

Each call takes the HBRWS copybook-defined structure as a parameter.

7.6.1 HBRWS header structure

The HBRWS header structure is required for all zRule Execution Server for z/OS client API calls. It is in the `<INSTALL-ROOT>.SHBRCOBS` data set. It must be included in any client programs that call zRule Execution Server for z/OS. Example 7-11 shows the layout of the structure.

Example 7-11 Layout of the structure

```
01 HBRA-CONN-AREA.
    10 HBRA-CONN-EYE                PIC X(4)   VALUE 'HBRC'.
    10 HBRA-CONN-LENTH              PIC S9(8)  COMP  VALUE +3536.
    10 HBRA-CONN-VERSION            PIC S9(8)  COMP  VALUE +2.
    10 HBRA-CONN-RETURN-CODES.
       15 HBRA-CONN-COMPLETION-CODE PIC S9(8)  COMP  VALUE -1.
       15 HBRA-CONN-REASON-CODE     PIC S9(8)  COMP  VALUE -1.
    10 HBRA-CONN-FLAGS              PIC S9(8)  COMP  VALUE +1.
    10 HBRA-CONN-INSTANCE.
       15 HBRA-CONN-PRODCODE        PIC X(4)   VALUE SPACES.
       15 HBRA-CONN-INSTCODE        PIC X(12)  VALUE SPACES.
       15 HBRA-CONN-SSID            PIC X(4)   VALUE SPACES.
       15 HBRA-CONN-GRPID           PIC X(4)   VALUE SPACES.
    10 HBRA-RESERVED01              PIC S9(8)  COMP  VALUE 0.
    10 HBRA-RESERVED02              PIC S9(8)  COMP  VALUE 0.
    10 HBRA-RESERVED03              PIC S9(8)  COMP  VALUE 0.
    10 HBRA-CONN-RULE-CCSID         PIC S9(8)  COMP  VALUE 0.
    10 HBRA-CONN-RULEAPP-PATH       PIC X(256) VALUE SPACES.
    10 HBRA-RESPONSE-AREA                      VALUE SPACES.
       15 HBRA-RESPONSE-MESSAGE     PIC X(1024).
    10 HBRA-RA-INIT                            VALUE LOW-VALUES.
       15 HBRA-RESERVED04           PIC X(1792).
    10 HBRA-RA-PARMETERS
       REDEFINES HBRA-RA-INIT.
       15 HBRA-RA-PARMS  OCCURS 32.
          20 HBRA-RA-PARAMETER-NAME     PIC X(48).
          20 HBRA-RA-DATA-ADDRESS       USAGE POINTER.
          20 HBRA-RA-DATA-LENGTH        PIC 9(8) BINARY.
    10 HBRA-RESERVED.
      15 HBRA-RESERVED05            PIC X(12).
      15 HBRA-RESERVED06            PIC X(64).
      15 HBRA-RESERVED07            PIC X(64).
```

```
    15 HBRA-RESERVED08          PIC X(128).
    15 HBRA-RESERVED09          PIC X(132).
```

The entire HBRA-CONN-AREA must be passed on each of the three API calls to zRule Execution Server for z/OS. We describe the important elements of this structure in the following sections.

HBRA-CONN-RETURN-CODES

This element provides response and reason codes to the requested API call. If these responses do not have a zero value on their return from an API, the user documentation provides more details about the error that occurred.

HBRA-CONN-RULEAPP-PATH

This element is important for the data structure. This value is used by zRule Execution Server for z/OS to identify which decision to execute on the specific data. After a decision is deployed to zRule Execution Server for z/OS, the value for the RULEAPP-PATH can be identified by logging on to the administrator console and viewing the deployed decision.

The RULEAPP-PATH uses this structure:

/<RULEAPP-NAME>/<RULEAPP-VER>/<RULESET-NAME>/<RULESET-VER>

The RULE-APP-VER and RULESET-VER are generally required only if the client wants to execute a particular version of a decision that is deployed to zRule Execution Server for z/OS. In most cases, the client wants to execute the latest version of the decision. In this case, the path is simplified to this structure:

/<RULEAPP-NAME>/<RULESET-NAME>

The Miniloan sample that is supplied with the product uses this structure:

```
/zRulesMiniLoanDemoRuleApp/zRulesMiniLoanDemo
```

HBRA-RESPONSE-AREA

If any text messages or warnings are returned by the Java portion of the server, they are returned in this data area to help you diagnose any problems. A preferred practice is to write this area out to an application log, if a nonzero return code is received.

HBRA-RA-PARAMETERS

This section is required only on the HBRRULE API call and provides the user data that is used to evaluate the decision. The structure is a list of triplets to pass the parameter name, a pointer to the data in working storage, and the length of the data.

HBRA-PARAMETER-NAME

This name is the name of the parameter as it is known to the decision. This parameter name is defined as part of the rule-authoring process. The parameter name is case-sensitive and must be added as a character string, exactly as it is defined in the ruleset.

HBRA-DATA-ADDRESS

This element is a pointer to the location of the parameter in your working storage. Normally, this element points to the address of the 01-level element in your copybook that you imported to define this parameter to the decision. It is normally set like the following example:

```
set HBRA-DATA-ADDRESS(1) to address of Borrower
```

HBRA-DATA-LENGTH

This element defines the length of the data structure to which the HBRA-DATA-ADDRESS points. Ensure that you set this element correctly so that all necessary data is passed across to the decision execution. You can use COBOL-intrinsic functions to calculate this value, for example:

```
move LENGTH OF Borrower to HBRA-DATA-LENGTH(1)
```

When the data definition depends on the table structure, ensure that the supplied length of the data structure is calculated assuming the maximum length of any tables.

7.6.2 HBRCONN API call

You use the HBRCONN API call to establish a connection to the zRule Execution Server for z/OS server. The HBRA-CONN-AREA data structure is passed as a parameter to this call. In CICS, when using a zRule Execution Server for z/OS stand-alone server or the locally optimized Java virtual machine (JVM) server deployment, the connection call is made at startup or when the HBRC transaction is run, not during the HBRCONN call.

7.6.3 HBRRULE API call

The HBRRULE API call actually invokes the decision for evaluation. The HBRA-CONN-AREA data structure is passed as a parameter and must contain references to the ruleset parameter data that is required for evaluating the decision.

Multiple HBRRULE calls can be made within a single HBRCONN/HBRDISC pair.

7.6.4 HBRDISC API call

You use the HBRDISC API call to disconnect from the server after all decisions are evaluated for this application. This call also takes the HBRA-CONN-AREA parameter. In CICS, the HBRDISC does not disconnect this CICS region explicitly from a Standalone zRule Execution Server for z/OS or a locally deployed zRule Execution Server for z/OS, but only this instance of an application.

7.7 Authoring considerations for performance

When authoring the rules, consider the implications on the performance of the decision. *Proven Practices for Enhancing Performance: A Q & A for IBM WebSphere ILOG BRMS 7.1*, REDP-4775, provides guidance about configuring and authoring for performance.

8

Runtime configuration options for Decision Server on z/OS

Multiple runtime environments are possible on z/OS. This chapter describes those environments for zRule Execution Server for z/OS and how to configure them. This chapter contains the following sections:

- Overview
- Running on z/OS stand-alone
- Configuring the Standalone zRule Execution Server for z/OS (one zRule Execution Server for z/OS/one console)
- A single RES console managing multiple zRule Execution Server for z/OS instances on one LPAR
- Configuring the zRule Execution Server for z/OS running on a CICS JVM server

© Copyright IBM Corp. 2012. All rights reserved.

8.1 Overview

The runtime environments on z/OS all support the COBOL execution object model (COBOL XOM), including the Rule Execution Server running on WebSphere Application Server on z/OS. The COBOL XOM capability is not supported by the rule engine running on WebSphere Application Server on a distributed platform.

Figure 8-1 shows the runtime environments that can be configured on z/OS:

- zRule Execution Server for z/OS hosted on CICS
- Standalone zRule Execution Server for z/OS
- Rule Execution Server hosted on WebSphere Application Server

All these run times support the COBOL XOM capability.

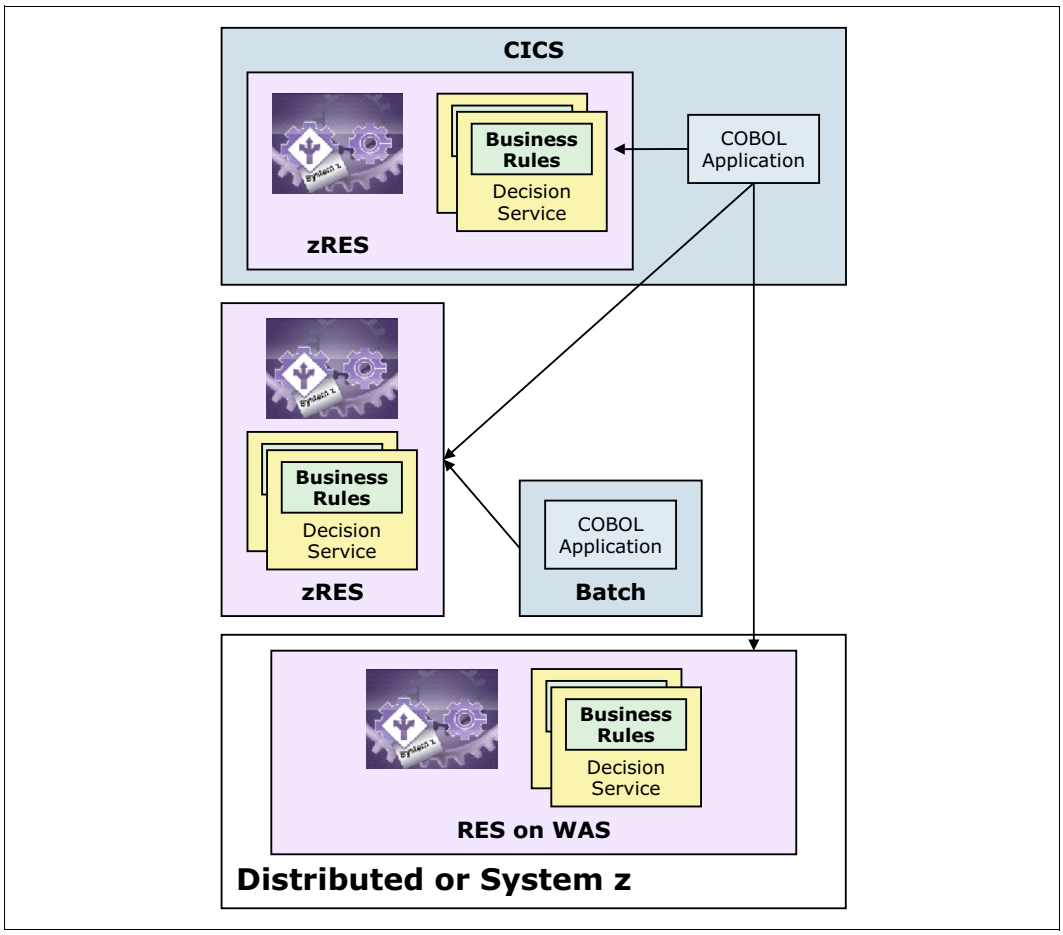


Figure 8-1 Runtime configurations

8.1.1 Configuring the run times

You set up each runtime configuration by changing the related parameter values that are grouped into a number of partitioned data set (PDS) members, as listed in Table 8-1 on page 225.

Table 8-1 Configuration parameters

Member name	Description
HBRBATCH	Used by the client to connect with the server
HBRCICSD	DB2 on CICS
HBRCICSJ	Used for CICS setup
HBRCICSZ	Execution server CICS connects with
HBRCMMN	Common parameters between zRule Execution Server for z/OS and Console
HBRCNSL	Console parameters
HBRDB2	DB2 connection information for Console and zRule Execution Server for z/OS
HBRFILE	File persistence for proof of concept only
HBRINST	Customer configuration values
HBRMSTR	Server parameters
HBRSCEN	Input for the Miniloan sample

Refer to the product documentation for a description of the parameters in these members:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/index.jsp

Each configuration parameter is updated with the site-specific values when the following job is used to configure a zRule Execution Server for z/OS instance.

Use the job HBRUUPTI to help configure a zRule Execution Server for z/OS instance. This job takes the customer values from the member HBRINST and stamps them into the related configuration parameter members and configuration JCL jobs for that runtime instance.

8.1.2 Prerequisite checklist

Use Table 8-2 on page 226 to check that the z/OS system is at the correct prerequisite level.

Table 8-2 Prerequisites

Item	Value
z/OS level	z/11
Java level + service	6.0.1 (64-bit support)
DB2 + service	9.1
CICS + service	CICS 4.1 + UK57632, UK69637, UK69654, and UK9655 or CICS 4.2 However, you can use CICS 3.*x* to run programs that connect to a stand-alone zRule Execution Server for z/OS.
Logical partition (LPAR) environment	
zaap/ziip java/db2 workload	
WebSphere Application Server level	V7 with Fix Pack 17
WebSphere Operational Decision Management	HDM7500 and HDM7501 (mandatory) HDM7502 - Rule Runtime Components for Decision Server JDM7503 - Event Runtime Components for Decision Server DM7504 - Decision Center

8.2 Running on z/OS stand-alone

This section describes how to run the rule engine on z/OS to consume batch work.

8.2.1 Standalone zRule Execution Server for z/OS

Figure 8-2 shows the runtime environment of a stand-alone zRule Execution Server for z/OS.

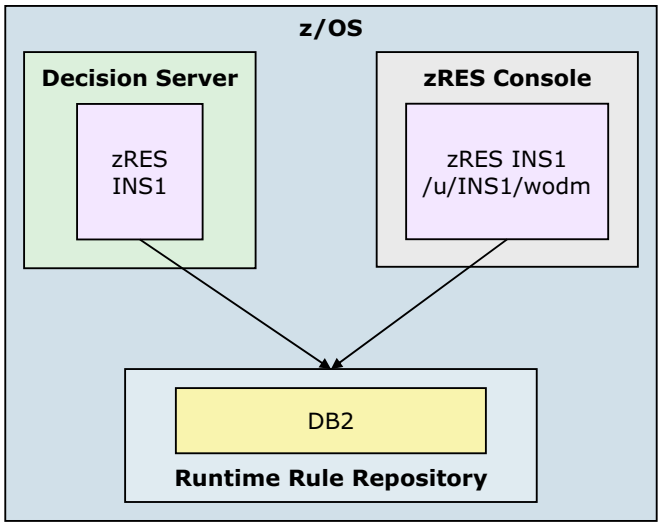


Figure 8-2 zRes Standalone

8.2.2 Configuring the Standalone zRule Execution Server for z/OS

To configure the Standalone zRule Execution Server for z/OS, you must edit the values in the HBRINST member and run the HBRUUPTI job using the HBRINST member as input to this job. This action creates a set of data sets that are configured for this zRule Execution Server for z/OS instance.

For example, the output from HBRUUPTI might produce the runtime configuration data sets that are shown in Table 8-3 and Example 8-1 for one zRule Execution Server for z/OS instance.

Table 8-3 Instance configuration data sets

Data set	Description
….SHBRJCL	Instance configuration jobs
…..SHBRPARM	Instance configuration values
…..SHBRPROC	Configured zRule Execution Server for z/OS started tasks
…..SHBRWASC	WebSphere Application Server configuration

Example 8-1 Example runtime configuration data sets

```
++HBRWORKDS++..INST1.SHBRJCL
++HBRWORKDS++.INST1.SHBRPARM
++HBRWORKDS++.INST1.SHBRPROC
++HBRWORKDS++.INST1.SHBRWASC
```

This step is repeated for each zRule Execution Server for z/OS instance that you create. Each zRule Execution Server for z/OS has a unique identifier that is given by the HBRSSID value.

Defining the zRule Execution Server for z/OS instance working directory

Each zRule Execution Server for z/OS instance, as well as having a set of data sets, has a file system location. The run time, when running, uses the file system location, which is called a *working directory*. Running the HBRCRTI job from the instance data sets SHBRJCL sets up the working directory in z/OS UNIX System Services for the zRule Execution Server for z/OS instance.

8.2.3 Creating the data sets that are changed for the zRule Execution Server for z/OS instance

This section describes how to create the data sets that are changed for the zRule Execution Server for z/OS instance.

Customizing the HBRINST member of SHBRPARM

Using the following tables, Table 8-4 on page 228, Table 8-5 on page 229, Table 8-6 on page 230, Table on page 232, and 8.2.4, “Creating the Working Datasets using HBRUUPTI” on page 233, to customize the HBRINST data set to your system environment. Use the second column to record your values.

Preferred practice: The preferred practice for this section is to copy the target library SHBRPARM PDS member HBRINST into a new location outside of the target library. Typically, you create a PDS called HLQ.CONFIG (where HLQ is the product high-level qualifier). Then, when creating an instance of zRule Execution Server for z/OS, a copy of the HBRINST member is created to track all server changes. So, for example, if the first instance is HBR1, the data set created is HBR1INST.

When running the customization job, SHBRJCL(HBRUUPTI), the INLINES data definition (DD) card must point to this new location of the HBRINST member:

```
000033 //INLINES DD DISP=SHR, DSN=&HBRHLQ, CONFIG_(HBR1INST)
```

At the time of writing this book, a known defect existed on this job. When you run this job with the new HBRINST, the output Working Datasets do not show the updated values in WORKDS.SHBRPARM(HBRINST).

Table 8-4 HBRINST customization values for rules on z/OS

Value	Your chosen value	Example value	Reason to update, change, or leave the default
++HBRSSID++		HBR1	Every time that a new zRule Execution Server for z/OS is set up, modify this value. Setting a naming convention that can scale with your system is key.
++HBRHLQ++		HBR.V750	This value is the installation target library from the product installation.
++HBRINSTPATH++		/usr/lpp/zDM/V7R5M0	This value is the root installation directory for the IBM WebSphere Operational Decision Management product in z/OS UNIX System Services.
++HBRWORKPATH++		/u/HBR1	This value is the work path for the specific instance of the server. This value must be updated for each new zRule Execution Server for z/OS instance.
++HBRWORKDS++		HBR.WORKDS.HBR1	For each new instance of the server, this value must be updated to a value that references the new instance. This value is the data set name for the changed output from the HBRUUPTI job.
++HBRJAVAHOME++		/java/java/601_but64_GA/J6.0.1_64	This value is the *root* location of Java 6.0.1 on z/OS in UNIX System Services.
++HBRCONSOLEPORT++		34114	This value is the port that is used for the zRule Execution Server for z/OS Execution Server Console. This value is the port you use to deploy and view deployed artifacts.
++HBRCONSOLECOMPORT++		44114	This value is the port that is used by the zRes Console and zRule Execution Server for z/OS instance to communicate with each other.

Table 8-5 HBRINST customization values for CICS Java virtual machine (JVM) server

Value	Your chosen value	Example value	Reason to update, change, or leave the default
++CICSWORKPATH++		/u/IYGBNCA8	This value is the CICS UNIX System Services working directory path. We set up each server so that each new zRule Execution Server for z/OS instance has its own CICS work path. This way, problem determination is easier with separate logs for each zRule Execution Server for z/OS instance.
++CICSHLQ++		CTS420.CICS	This value is the HLQ for the CICS installation. Change this value to match the CICS installation HLQ.
++CICSCSDSN++		CTS420.APPLID.DFHCSD	This value is the HLQ for the CICS region CICS system definition data set (CSD) file. For each new region into which a zRule Execution Server for z/OS is to be deployed, this value must be updated.
++CICSLIST++		DFHLIST	This value is the name of the CICS list to which HBRGROUP is added. With each region change, this value must be updated, if necessary.
++HBRJAVA31HOME++		/java/java601_bit31_GA/J6.0.1	If the system is installed within a CICS V4.1 region, the 31-bit JVM must be set here. If the CICS region is CICS V4.2, the zRule Execution Server for z/OS that is set up in it does not need this component.
++JDBCPLAN++		DSNJCC	This value is the plan used for Java Database Connectivity (JDBC) connections and CICS.

Table 8-6 HBRINST customization values for DB2 variable changes

Value	Your chosen value	Example value	Reason to update, change, or leave the default
++DB2HLQ++		SYS2.DB2.V9R1	This value is the HLQ of the DB2 installation.
++DB2RUNLIB++		DSN910GP.RUNLIB.LOAD	The DB2 Run library location.
++DB2SUBSYSTEM++		db2_subsystem_id	This value is the subsystem name.
++DB2VCAT++		DSN910GP	The DB2 integrated catalog facility (ICF) catalog.
++DB2CURRSQLID++		ZILOGDB	The current SQL ID. The owner of a table space, database, or storage group. An authorization ID with the same name as a schema implicitly has CREATIN, ALTERIN, and DROPIN privileges for that schema.
++RESDATABASE++		RESDB1	The name of the database that is used by the zRule Execution Server for z/OS instance.
++RTSDATABASE++		RTSDB1	The name of the database that is used by the Decision Center instance.
++RESSTOGROUP++		RESSTG1	The name of the storage group that is used by the zRule Execution Server for z/OS instance.
++RTSSTOGROUP++		RTSSTG1	The name of the storage group that is used by the Decision Center instance.
++DB2TABLEBP++		BP1	The buffer pool name for the tables.
++DB2INDEXBP++		BP2	The buffer pool name for the indexes.
++DB2LOBBP++		BP3	The buffer pool name for large objects.
++DB2SAMPLEPROGRAM++		DSNTEP2	The DB2 program name.
++DB2SAMPLEPROGRAMPLAN++		DSNTEP91	The DB2 plan name.
++DB2USER		ZILOGDB	The user ID for accessing the DB2 database.
++DB2PSWD++		your_pwd	The password for accessing the DB2 database.

Value	Your chosen value	Example value	Reason to update, change, or leave the default
++DB2CONSTR++		host.db2.ipaddr.com:49100/DSN910GP	The connection string for a JDBC Universal Driver Type 4 connection, with the format: *ipaddress:portnumber/database_subsystem_location*
++DB2JARLOCN++		/usr/lpp/db2910/classes	The location of the DB2 classes.

Table 8-7 HBRINST customization values for WebSphere Application Server

Value	Your chosen value	Example value	Reason to update, change, or leave the default
++WASINSTPATH++		/WebSphere/V70IL2Z1/AppServer	The installation directory of WebSphere Application Server.
++WAS_HOME++		/WebSphere/V70IL2Z1/AppServer	The WebSphere Application Server home directory. It is unique for each server instance.
++SECURITYTYPE++		RACF®	Set to RACF if your WebSphere Application Server system is configured to use RACF. Set to WebSphere Application Server if your WebSphere Application Server system is configured to use federated security.
++DMGRPATH++		/WebSphere/V7ILGDM	The DManager Path in a network deployment environment. Use this variable if the path is too long for the ++DMGRPATH++ variable. You can split the path over the two variables: ++DMGRPATH++ and ++DMGRPATH2++.
++RESWASINSTANCE++		/u/HBR1/was_server1	The location of the `res.property` file.
++RTSWASINSTANCE++		/u/HBR1/was_server1	The location of the `rts.property` file.

Value	Your chosen value	Example value	Reason to update, change, or leave the default
++DVSWASINSTANCE++		/u/HBR1/was_server1	The location of the `dvs.property` file.
++WASSERVERNAME++		Serveril2Base	The WebSphere Application Server server instance name.
++PROFILE++		default	The WebSphere Application Server profile. It is set to default on z/OS.
++CELLNAME++		cell01	The WebSphere Application Server cell name.
++NODENAME++		node1	The WebSphere Application Server server node name.
++ADMINUSER++		wasadmin	The WebSphere Application Server server administration user ID.
++ADMINPSWD++		admin_pwd	The WebSphere Application Server server administration password.
++WASVERSION++			The WebSphere Application Server version number.
++J2CID++		ZILOGDB	The J2EE Connector architecture (J2C) authentication alias user ID.
++EJBHLQ++		CLID	The System Authorization Facility (SAF) prefix for EJBROLEs. It might be blank.

Table 8-8 HBRINST customization values for section after WSADMIN-specific install script parameters

Value	Your chosen value	Example value	Reason to update, change, or leave the default
++DB2NATIVELOC++		/usr/lpp/db2910/lib	The location of the DB2 native library files.
++RTSDRIVERTYPE++			The JDBC Universal Driver Type for the request to send (RTS) data source connection.
++XOMDRIVERTYPE++			The JDBC Universal Driver Type for the XOM data source connection.
++RESDRIVERTYPE++			The JDBC Universal Driver Type for the RES data source connection
++DB2SERVNAME++		host.db2.ipaddr.com	The DB2 host name or IP address.
++DB2PORT++		49100	The DB2 connection port.

8.2.4 Creating the Working Datasets using HBRUUPTI

The HBRUUPTI member that is within the ++HBRHLQ++.SHBRJCL data set uses the values in the HBRINST member to populate the SHBRJCL, SHBRPARM, SHBRPROC, and SHBRWASC data sets that are changed to your system environment.

Changing HBRUUPTI

You must perform several steps to change HBRUUPTI to create the new Working Datasets for the zRule Execution Server for z/OS server. Customize the HBRINST data set to your system's environment.

The preferred practice is to copy the target library SHBRPARM member HBRINST into a new location outside of the target library. Typically, a new PDS is created called <*hlq*>.CONFIG (where <*hlq*> is the product high-level qualifier). Then, you have a copy of the HBRINST member to track all server changes for each new instance of zRule Execution Server for z/OS.

Preferred practice: If using the preferred practice for the installation, update the INLINES DD card with the member for the current configuration. So, for example, if the first instance is HBR1, the member that is created is HBR1INST.

When running the customization job, SHBRJCL(HBRUUPTI), the HBRINST field update to the INLINES DD card must be changed to reflect this new location of the HBRINST file:

```
000033 //INLINES DD DISP=SHR, DSN=&HBRHLQ, CONFIG_(HBR1INST)
```

When you run this job with the new HBRINST, the output Working Dataset member does not show the updated values in the member HBRINST, but all the other members are modified to the user's values.

Changing HBRUUPTI

To change HBRUUPTI, follow these steps:

1. Update HBRUUPTI PROC HBRHLQ=++HBRHLQ++ with the value of your HBRHLQ from Table 8-4 on page 228. Figure 8-3 shows this update line.

```
000029 //HBRUUPTI PROC HBRHLQ=++HBRHLQ++
```

Figure 8-3 HBRUUPTI PROC update

2. Update the -INFILE- and -OUTFILE- with ++HBRHLQ++ and ++HBRWORKDS++ on HBRUUPTI.INPUT for SHBRJCL. See Figure 8-4 for the unchanged version.

```
000038 //HBRUUPTI.INPUT DD *
000039  -INFILE-     ++HBRHLQ++.SHBRJCL
000040  -OUTFILE-    ++HBRWORKDS++.SHBRJCL
```

Figure 8-4 Step 2 of HBRUUPTI update

3. Repeat this action three times for the files for SHBRPROC, SHBRPARM, and SHBRWASC by updating the -INFILE- and -OUTFILE- with ++HBRHLQ++ and ++HBRWORKDS++. Figure 8-5 shows these updates. This step is the last part of the changes for the SHBRPROC, SHBRPARM, and SHBRWASC.

```
//HBRUUPTI.INPUT DD *
 -INFILE-      ++HBRHLQ++.SHBRPROC
 -OUTFILE-     ++HBRWORKDS++.SHBRPROC
/*
//PARM  EXEC HBRUUPTI
//HBRUUPTI.SYSTSIN DD *
 %HBRUUPTE
//HBRUUPTI.INPUT DD *
 -INFILE-      ++HBRHLQ++.SHBRPARM
 -OUTFILE-     ++HBRWORKDS++.SHBRPARM
/*
//WASC  EXEC HBRUUPTI
//HBRUUPTI.SYSTSIN DD *
 %HBRUUPTE
//HBRUUPTI.INPUT DD *
 -INFILE-      ++HBRHLQ++.SHBRWASC
 -OUTFILE-     ++HBRWORKDS++.SHBRWASC
/*
```

Figure 8-5 HBRUUPTI step 3 customization

4. Submit the job to create the Working Datasets for the server instance. This job creates the following data sets:
   - ++HBRWORKDS++.SHBRJCL
   - ++HBRWORKDS++.SHBRPARM
   - ++HBRWORKDS++.SHBRPROC
   - ++HBRWORKDS.SHBRWASC

8.2.5 Creating the working directories in UNIX System Services

After submitting the HBRUUPTI job, go to the ++HBRWORKDS++.SHBRJCL PDS, and open the job HBRCRTI. This job runs an `hbrcrtin.sh` script that is located in the ++HBRINSTPATH++ that was set in Table 8-4 on page 228. This job creates the ++HBRWORKPATH++ directory in UNIX System Services, which creates the following directories: `config`, `logs`, `res_data`, `res_xom`, and `work`. Figure 8-6 shows these directories.

```
_ Dir          8192  config
_ Dir          8192  logs
_ Dir          8192  res_data
_ Dir          8192  res_xom
_ Dir          8192  work
```

Figure 8-6 Directories created by ++HBRWORKDS++.SHBRJCL(HBRCRTI)

8.3 Configuring the Standalone zRule Execution Server for z/OS (one zRule Execution Server for z/OS/one console)

For every type of setup, you must initially configure one Standalone zRule Execution Server for z/OS with one console. This configuration is required when setting up the other configurations that are described later in this chapter. Those sections refer to completing this configuration before setting up the other types of environments.

8.3.1 Defining a new subsystem for zRule Execution Server for z/OS

The first step for the configuration of the Standalone zRule Execution Server for z/OS is to define the subsystem in which the new instance runs. The systems programmer must perform this task. Figure 8-7 shows the following command that must be run, where ++HBRSSID++ is the Subsystem ID that was set in Table 8-4 on page 228:

```
SETSSI ADD,SUBNAME=++HBRSSID++
```

```
                    System Command Extension

Type or complete typing a system command, then press Enter.

===> SETSSI ADD,SUBNAME=++HBRSSID++
===>
```

Figure 8-7 Setting the zRule Execution Server for z/OS instance subsystem name

8.3.2 Creating the started tasks (HBRXCNSL and HBRXMSTR)

The next task in this configuration is to change HBRXCNSL and HBRXMSTR to match the environment that you are setting up. When using the DB2 persistence layer, you must change HBRXCNSL and HBRXMSTR to reflect that you are using the DB2 persistence layer. These data sets are then copied to PROCLIB and defined as started tasks so that the Standalone zRule Execution Server for z/OS can be started for use in the z/OS environment.

Changing HBRXMSTR

Change the following JCL in HBRXMSTR, which is located in ++HBRWORKDS++.SHBRPROC(HBRXMSTR) from DSN=&HBRWORKD..SHBRPARM(HBRFILE) to DSN=&HBRWORKD..SHBRPARM(HBRDB2). Figure 8-8 on page 237 shows the JCL to change.

```
000020 //++HBRSSID++MSTR EXEC PGM=HBRMAIN,REGION=&REG
000021 //STEPLIB  DD DISP=SHR,DSN=&HBRHLQ..SHBRAUTH
000022 //SYSOUT   DD SYSOUT=&OUTCLAS
000023 //SYSABEND DD SYSOUT=&OUTCLAS
000024 //HBRLEOUT DD SYSOUT=&OUTCLAS
000025 //HBRPRINT DD SYSOUT=&OUTCLAS
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRMSTR)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRFILE)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-8 HBRXMSTR line to update from HBRFILE to HBRDB2 when adding DB2 persistence

The highlighted line in Figure 8-9 shows the change.

```
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRMSTR)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRDB2)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-9 HBRXMSTR line updated to use DB2 persistence

Now, the zRes server uses the database as its persistent store, from which it gets the rulesets to use at run time.

Changing HBRXCNSL

Change the following JCL in HBRXCNSL, which is located in ++HBRWORKDS++.SHBRPROC(HBRXCNSL) from DSN=&HBRWORKD..SHBRPARM(HBRFILE) to DSN=&HBRWORKD..SHBRPARM(HBRDB2). Figure 8-10 on page 238 show the line to change.

```
000019 //*************************************************
000020 //++HBRSSID++CNSL  EXEC PGM=HBRXMAIN,REGION=&REG
000021 //STEPLIB  DD DISP=SHR,DSN=&HBRHLQ..SHBRAUTH
000022 //SYSOUT   DD SYSOUT=&OUTCLAS
000023 //SYSABEND DD SYSOUT=&OUTCLAS
000024 //HBRLEOUT DD SYSOUT=&OUTCLAS
000025 //HBRPRINT DD SYSOUT=&OUTCLAS
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRCNSL)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRFILE)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-10 HBRXCNSL line to update from HBRFILE to HBRDB2 when adding DB2 persistence

Figure 8-11 shows the line that changed.

```
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRCNSL)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRDB2)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-11 HBRXCNSL line updated to use DB2 persistence

Production usage: You can use the file system setup in proofs of concept and the initial configuration. However, for production, the configuration must have a DB2 persistence layer.

Adding HBRXMSTR and HBRXCNSL to SYS1.PROCLIB

The next task is to copy HBRXMSTR and HBRXCNSL to SYS1.PROCLIB (or a similar PROCLIB on your environment). When copying over the data sets, the names change from HBRXMSTR to ++HBRSSID++MSTR and from HBRXCNSL to ++HBRSSID++CNSL, when the new tasks are started. Follow these steps:

1. Copy the ++HBRWORKDS++.SHBRPROC(HBRXMSTR) to SYS1.PROCLIB(++HBRSSID++MSTR).
2. Copy the ++HBRWORKDS++.SHBRPROC(HBRXCNSL) to SYS1.PROCLIB(++HBRSSID++CNSL).

SYS1.PROC: SYS1.PROCLIB is the default, so change it for your environment. ++HBRSSID++ was set in Table 8-4.

Authorizing the server instance as a started task

Authorize both ++HBRSSID++MSTR and ++HBRSSID++CNSL as started task procedures executing with the same user ID. Use the following commands:

```
RDEFINE STARTED ++HJBRSSID++MSTR.* STDATA(USER(<HBRSSID_USER>)
GROUP(<HBRSSID_GROUP>)
RDEFINE STARTED ++HJBRSSID++CNSL.* STDATA(USER(<HBRSSID_USER>)
GROUP(<HBRSSID_GROUP>)
```

HBRSSID_USER is the server user ID. *HBRSSID_GROUP* is the RACF security group name that is provided to you by your security administrator.

The started task definitions

Whether starting the Standalone zRule Execution Server for z/OS started task or the CICS zRule Execution Server for z/OS started task, both started tasks require that the configuration parameters are provided to the job. The parameters are provided by the DD card HBRENVPR on each started task. The DD card HBRENVPR specifies the input parameter members.

8.3.3 Securing the zRule Execution Server for z/OS for z/OS resources

With IBM WebSphere Operational Decision Management, you can secure the resources, files, and functions with RACF. In this section, we describe how to create this security for the server using RACF.

Security options

If running the zRule Execution Server for z/OS in production, you might want to secure all or part of the zRule Execution Server for z/OS resources. However, if you plan to run the server in a testing environment, perhaps you want security disabled. You can use the following options for security.

Within the file system, you can secure the following resources:

- The working directory so that only the authorized user IDs can access internal data
- The installation directory so that only the authorized user IDs can access the files that are needed to run the server

Using RACF, you can secure the following resources:

- You can secure the server resources that you use to perform the following tasks:
  - Issue zRule Execution Server for z/OS commands from the z/OS console (or equivalent)
  - Sign on to the Rule Execution Server Console
  - Connect to the zRule Execution Server for z/OS to execute rulesets
- You can secure a subset of server resources.

  For example, you can secure access to the Rule Execution Server Console only.

Securing access to the working directory and installation directory

The working directory contains data that includes logs from the zRule Execution Server for z/OS, component details, and trace output. The installation directory contains the necessary files to run the zRule Execution Server for z/OS server. The server user ID needs to read and execute access for ++HBRWORKPATH++, ++HBRINSTPATH++, the zRule Execution Server for z/OS Work Path, and the IBM WebSphere Operational Decision Management installation directory, if the permissions are to be changed on these directories.

Creating the RACF Classes for securing server resources

You can manage zRule Execution Server for z/OS using RACF classes. We must create the three RACF classes through the use of the ++HBRWORKDS++.SHBRJCL(HBRCRECL) job. To secure the resources for the zRule Execution Server for z/OS instance, your RACF administrator must run the HBRCRECL job.

Creating the RACF classes

Using RACF, you can secure the following information:

- Ask the RACF Administrator to run the HBRCRECL job or extract the code to use the preferred execution methods to perform the following tasks:
  - Issue zRule Execution Server for z/OS commands from the z/OS console (or equivalent)
  - Sign on to the Rule Execution Server Console
  - Connect to the zRule Execution Server for z/OS to execute rulesets
- You can secure a subset of server resources.

  For example, you can secure access to the Rule Execution Server Console only.

When your RACF administrator runs the HBRCRECL job, the job creates three RACF classes: HBRADMIN, HBRCONN, and HBRCMD. Table 8-9 explains the characteristics of these classes.

Table 8-9 RACF classes created by ++HBRWORKDS++.SHBRJCL(HBRCRECL)

Class	Description
HBRADMIN	This class controls whether server security and security for specific server resources are enabled or disabled.
HBRCONN	This class specifies the user IDs that are authorized to connect to the zRule Execution Server for z/OS and to execute rulesets. This class is ignored if server security is disabled.
HBRCMD	This class specifies the user IDs that are authorized to issue zRule Execution Server for z/OS commands, such as START, STOP, PAUSE, or RESUME from the z/OS console (or equivalent). This class is ignored if server security is disabled.

POSIT: The supplied JCL in HBRCRECL gives a POSIT value of 128. Change POSIT, as required, to match your security environment requirements.

After running the HBRCRECL job, give the server user ID read access to the class profile using the following commands:

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(<HBRSSID_USER>) ACCESS(READ)

SETROPTS RACLIST(FACILITY) REFRESH
```

In this example, <*HBRSSID_USER*> represents the server user ID, which is the ID under which the server runs.

Disabling types of security

In IBM WebSphere Operational Decision Management, you optionally can disable all types of security or parts of the security. When the HBRADMIN class is created, security is enabled on all zRule Execution Server for z/OS instances. Security can be enabled and disabled, as

required. On a test system, you might want no security on the instance so that you can test more freely, but you do not have to disable the security to all instances that are used.

To disable levels of security, you must apply separate profiles to the HBRADMIN class. Table 8-10 gives the profiles that can be added to the HBRADMIN class using the following commands:

```
RDEFINE HBRADMIN <RESOURCE_PROFILE> UACC(NONE)

SETROPTS RACLIST(HBRADMIN) REFRESH
```

Table 8-10 Resource profiles for the HBRADMIN RACF class to disable parts of security on the zRule Execution Server

Resource profile	Description
++HBRSSID++.NO.SUBSYS.SECURITY	This profile disables all security for a particular server instance. If server security is disabled, HBRCONN and HBRCMD classes are not used.
++HBRSSID++.NO.CONNECT.SECURITY	This profile disables connection security for a particular server instance, but it maintains other types of security.
++HBRSSID++.NO.RESCONSOLE.SECURITY	This profile disables console security for a particular server instance, but it maintains other types of security.
++HBRSSID++.NO.COMMAND.SECURITY	This profile disables command security for a particular server instance, but it maintains other types of security. If you disable command security, any user can issue a zRule Execution Server for z/OS command from the z/OS console.

The ++HBRSSID++ value is set in Table 8-4 on page 228 for the particular server instance to which this profile is set. For more information, refer to 8.3.4, “Starting the new instance” on page 244. If the RESCONSOLE profile is used, skip “Managing console security” on page 242. If the COMMAND profile is used, skip “Managing command security” on page 243, or if the CONNECT profile is used, skip “Managing connection security”.

Managing connection security

You set up connection security to ensure that only authorized user IDs can connect to the zRule Execution Server for z/OS instance to execute rulesets. Connection security uses the HBRCONN RACF class to authorize user IDs to connect to the server instance. If the profile ++HBRSSID++.NO.SUBSYS.SECURITY or ++HBRSSID++.NO.CONNECT.SECURITY is used, the HBRCONN class is ignored.

To implement connection security, you must authorize the user ID under which the server runs and the user IDs of any applications that execute rulesets. The following steps are required for authorizing user IDs to the HBRCONN class:

1. First, the resource profile needs the server instance defined to the HBRCONN class. Run the following command first to create the resource profile:

   ```
   RDEFINE HBRCONN ++HBRSSID++ UACC(NONE)
   ```

2. Give the server user ID UPDATE access to the ++HBRSSID++ resource profile. Use the following command:

   ```
   PERMIT ++HBRSSID++ CLASS(HBRCONN) ID(<HBRSSID_USER>) ACCESS(UPDATE)
   ```

UPDATE access: The server instance fails to initialize if the HBRCONN class does not have UPDATE access. This requirement does not affect a server instance with ++HBRSSID++.NO.SUBSYS.SECURITY or ++HBRSSID++.NO.CONNECT.SECURITY.

3. Refresh the ++HBRSSID++ resource profile using the following command:

```
SETROPTS RACLIST CLASS(HBRCONN) REFRESH
```

Next, you authorize the applications by using the following commands:

1. Give READ access to the ++HBRSSID++ resource profile to each user that you want to authorize. Use the following command:

```
PERMIT ++HBRSSID++ CLASS(HBRCONN) ID(<USER_ID>) ACCESS(READ)
```

User IDs: For batch jobs, <*USER_ID*> is the RACF user ID that is used by the batch job. For CICS transactions, <*USER_ID*> is the user ID that is assigned to the CICS address space.

2. Refresh the ++HBRSSID++ resource profile by using the following command:

```
SETROPTS RACLIST(HBRCONN) REFRESH
```

Managing console security

You use console security to ensure that there is control on the users that can access the zRule Execution Server for z/OS console. The zRule Execution Server for z/OS console security controls the ability to sign on to the zRule Execution Server for z/OS console. If security is enabled, users must enter a user ID and password to sign on.

If the profile ++HBRSSID++.NO.SUBSYS.SECURITY or the profile ++HBRSSID++.NO.RESCONSOLE.SECURITY is used, the HBRADMIN class is ignored.

A standard set of roles exists within the zRule Execution Server for z/OS that gives access rights to users. Enable console security by assigning user IDs to roles and then authorizing the roles to access the console.

Table 8-11 shows the profiles and the roles that they represent. We list the roles in order of increasing authority. RESMON is the lowest authority, and RESADMIN is the highest authority. ++HBRSSID++ is the ID of the subsystem where the server runs.

Table 8-11 zRule Execution Server for z/OS console security profiles

Resource profile	Role description
++HBRSSID++.ROLE.RESMON	Users with monitoring rights are only allowed to view and explore RuleApps, rulesets, decision services, Execution Units (XUs), and statistics. These users are not allowed to modify these entities. They can also select a trace configuration and view and filter trace information in Decision Warehouse. This authority applies only to Rule Execution Server on WebSphere Application Server for z/OS.
++HBRSSID++.ROLE.RESDEP	In addition to monitoring rights, users with deploying rights are allowed to deploy RuleApp archives, to edit and remove entities (RuleApps, rulesets, decision services, Java execution object module (XOM) resources, and libraries), and to run diagnostics.

++HBRSSID++.ROLE.RESADMIN	Users with administrator rights have full control over the deployed resources and access to information on the server. They can perform the following actions: ► Deploy, browse, and modify RuleApps, Java XOM resources, and libraries ► Monitor the decision history, purge the history, and back up the history ► Select a trace configuration, view and filter trace information, and clear trace information in Decision Warehouse ► Run diagnostics and view server information

Perform the following steps to enable console security:

1. Define each resource profile as shown in Table 8-11 on page 242 to the HBRADMIN class. Use the following commands to define all three roles:

```
RDEFINE HBRADMIN ++HBRSSID++.ROLE.RESMON UACC(NONE)
RDEFINE HBRADMIN ++HBRSSID++.ROLE.RESDEP UACC(NONE)
RDEFINE HBRADMIN ++HBRSSID++.ROLE.RESADMIN UACC(NONE)
```

2. Assign each user ID to one of the resource profiles using the following commands for the three roles:

```
PERMIT ++HBRSSID++.ROLE.RESMON UACC(NONE)
PERMIT ++HBRSSID++.ROLE.RESDEP UACC(NONE)
PERMIT ++HBRSSID++.ROLE.RESADMIN UACC(NONE)
```

3. Refresh the HBRADMIN class using the following command:

```
SETROPTS RACLIST(HBRADMIN) REFRESH
```

Managing command security

You use command security to ensure that only authorized users can issue zRule Execution Server for z/OS commands on the zRule Execution Server for z/OS console. Command security uses the HBRCMD RACF class to authorize user IDs to issue zRule Execution Server for z/OS commands.

If the profile ++HBRSSID++.NO.SUBSYS.SECURITY or the profile ++HBRSSID++.NO.COMMAND.SECURITY is used, the HBRCMD class is ignored.

When enabling command security on the zRule Execution Server for z/OS console, you must define a resource profile to the HBRCMD class for each command that you want to secure. Use the commands that are listed in Table 8-12 to secure the zRule Execution Server for z/OS console commands. ++HBRSSID++ is the ID of the subsystem where the server runs.

Table 8-12 zRule Execution Server for z/OS command security profiles

Resource profile	Command	Command description
++HBRSSID++.SET.TRACE	SET TRACE	Ability to turn trace on or off
++HBRSSID++.SET.RESCONSOLE	SET RES-CONSOLE	Start or stop the zRule Execution Server for z/OS console
++HBRSSID++.SET.PAUSE	PAUSE	Pause a job on the zRule Execution Server for z/OS
++HBRSSID++.SET.RESUME	RESUME	Resume a job on the zRule Execution Server for z/OS

To authorize users to issue zRule Execution Server for z/OS commands, execute the following commands:

- If you want to limit any of the commands in Table 8-12 on page 243 to authorized user IDs, you must define the resource profiles to the following HBRCMD class commands:

```
RDEFINE HBRCMD ++HBRSSID++.SET.TRACE UACC(NONE)
RDEFINE HBRCMD ++HBRSSID++.SET.RESCONSOLE UACC(NONE)
RDEFINE HBRCMD ++HBRSSID++.SET.PAUSE UACC(NONE)
RDEFINE HBRCMD ++HBRSSID++.SET.RESUME
```

- If you want to limit any of the commands in Table 8-12 on page 243 to authorized user IDs, you must define the resource profiles to the following HBRCMD class commands:

```
PERMIT ++HBRSSID++.SET.TRACE CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
PERMIT ++HBRSSID++.SET.RESCONSOLE CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
PERMIT ++HBRSSID++.SET.PAUSE CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
PERMIT ++HBRSSID++.SET.RESUME CLASS(HBRCMD) ID(<USER_ID>) ACCESS(UPDATE)
```

> **User IDs:** For batch jobs, <*USER_ID*> is the RACF User ID that is used by the batch job. For CICS transactions, <*USER_ID*> is the user ID that is assigned to the CICS address space.

- Refresh the HBRCMD class by using the following command:

```
SETROPTS RACLIST(HBRCMD) REFRESH
```

8.3.4 Starting the new instance

After completing the security setup and configurations, start the new server instance through the z/OS console.

Authorizing the load library

If you are setting up your first instance on the LPAR, you must authorize the load library. You must perform the following three commands:

1. Add the ++HBRHLQ++.SHBRAUTH load library to the authorized program facility (APF)-authorized libraries using the following command:

```
SETPROG APF,ADD,DSNAME=++HBRHLQ++.SHBRAUTH,SMS
```

2. Provide the RACF authorization for the ++HBRHLQ++.SHBRAUTH load library using the following two commands:

```
RALTER PROGRAM * ADDMEM('++HBRHLQ++.SHBRAUTH'//NOPADCHK)

SETROPTS WHEN(PROGRAM) REFRESH
```

   ++HBRHLQ++ is the product installation target library HLQ for the SHBRAUTH PDS.

Starting a server instance

To start a new server instance, issue the following command:

```
START ++HBRSSID++MSTR
```

If the server does not start, look at the output of the HBRMSTR job to see why it did not start. Typically, the server does not start for following reasons:

- The load library was not APF-authorized.
- ++HBRINSTPATH++ does not point to the proper UNIX System Services directory (if you go to the location that you put in the ++HBRINSTPATH++, it shows directories that include `IBM0`, `IBM1`, `IBM2`, `IBM3`, `IBM4`, `lib`, `shared`, and so on)

- If you use symbolic links on ++HBRINSTPATH++ or ++HBRWORKPATH++, these links might not link correctly. Therefore, you must verify the link.
- You did not execute the RACF security commands. Verify whether the RACF security commands were run by using the resource profile setup. Ensure that the commands executed and that the user starting the server is authorized to start the server.
- The ports that were used for the ++HBRCONSOLEPORT++ and ++HBRCONSOLECOMHOST++ were already in use by another application.
- The same port is used for ++HBRCONSOLEPORT++ and ++HBRCONSOLECOMHOST++.

8.3.5 Logging in to the zRule Execution Server for z/OS console and performing diagnostics

Now that the Standalone zRule Execution Server for z/OS console is up and running, you must run the diagnostics on it. You must use a RACF ID that is part of the RESADMIN group, which can run the diagnostics on the Rule Execution Server.

Go to `http:// ++HBRCONSOLECOMHOST++:++HBRCONSOLEPORT++/res`, and log in with the RACF user ID and password. Perform these steps:

1. When you are logged in, you see a console similar to the console that is shown in Figure 8-12. There are four tabs across the top. Click the **Diagnostics** tab.

Figure 8-12 Rule Execution Server Console Welcome panel

2. The Diagnostics View appears. Click **Run Diagnostics**, as shown in Figure 8-13 on page 246.

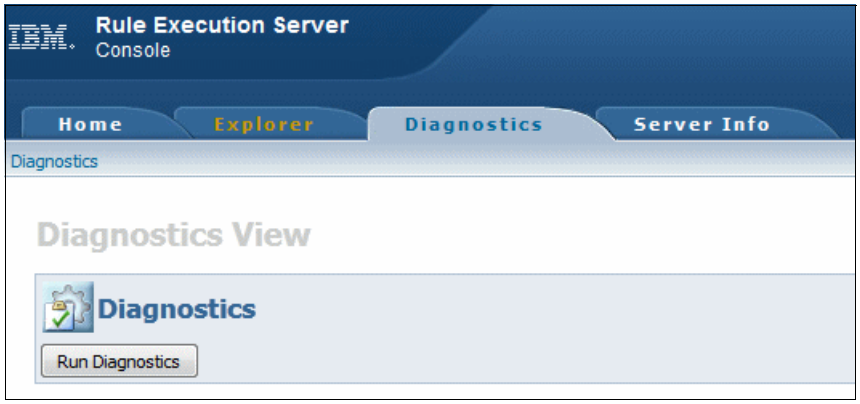


Figure 8-13 Rule Execution Server Diagnostics View

3. When you run the diagnostics, successful diagnostics show all results with green check marks. If there are any issues, the diagnostics tool helps you troubleshoot the problem. Figure 8-14. shows a successful deployment.

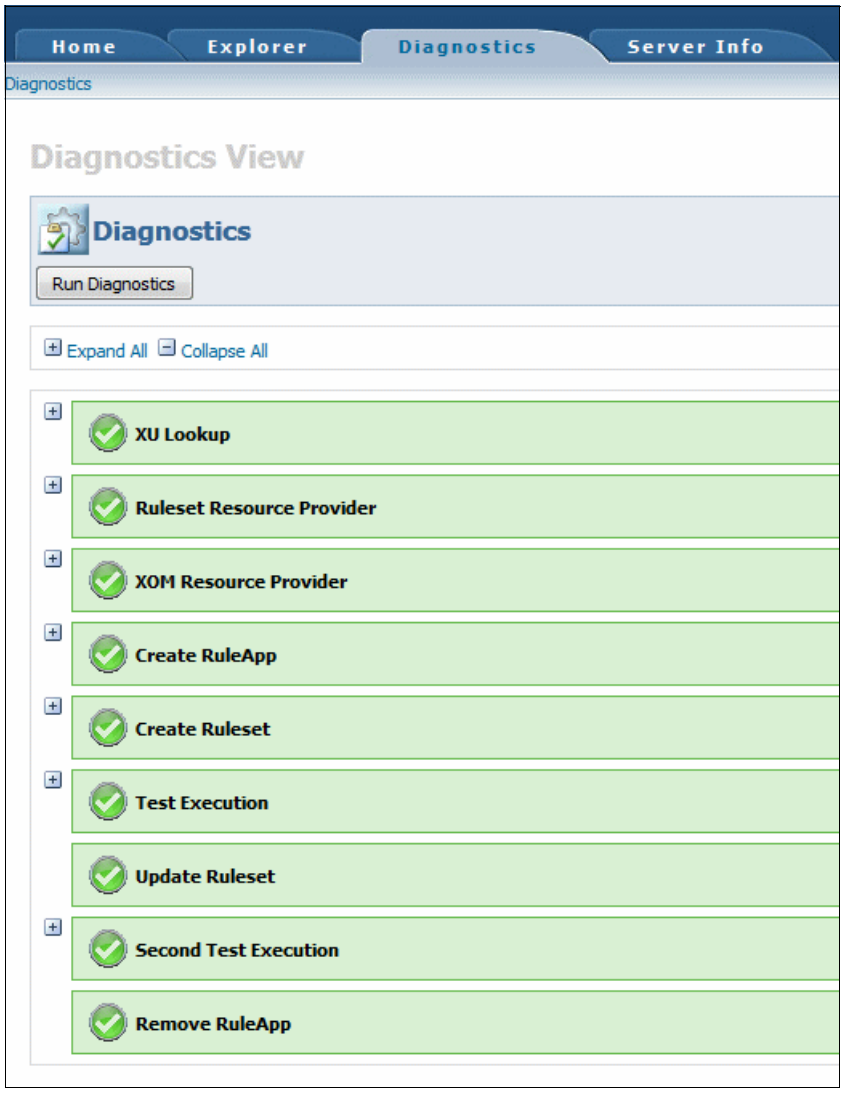


Figure 8-14 Rule Execution Server Diagnostics successful test

The Rule Execution Server is available for use. You can start using this instance of the zRule Execution Server for z/OS in your environment for rules execution.

8.4 A single RES console managing multiple zRule Execution Server for z/OS instances on one LPAR

Figure 8-15 shows one LPAR with two zRule Execution Server for z/OS batch servers that are administered by one console. The second zRule Execution Server for z/OS (INS2) is started with the parameter HBRCONSOLE=NO, so that INS2 does not launch another console.

The same HBRDB2 member is used by both zRule Execution Server for z/OS instances so that they connect to the same repository. The same HBRCMMN member can be used by each zRule Execution Server for z/OS instance. The value HBRCONSOLECOMPORT must be the same for both zRule Execution Server for z/OS instances, because both zRule Execution Server for z/OS instances communicate with the same zRule Execution Server for z/OS console.

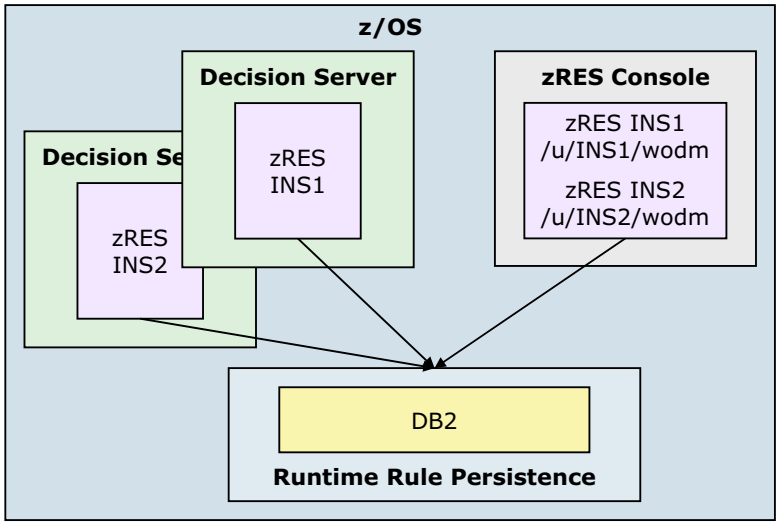


Figure 8-15 One console managing multiple Standalone zRule Execution Server for z/OS instances

8.4.1 Adding another zRule Execution Server for z/OS to an already running zRule Execution Server for z/OS console

Next, we describe an IBM WebSphere Operational Decision Management topology of adding a zRule Execution Server for z/OS server to a zRule Execution Server for z/OS console that is already running. We create this topology so that we can have multiple zRule Execution Server for z/OS servers running within a single LPAR.

8.4.2 Updating HBRINST to add a zRule Execution Server for z/OS to an existing zRule Execution Server for z/OS console

When setting up a new zRule Execution Server for z/OS, the process is similar to the process to set up the first zRule Execution Server for z/OS. In this section, we explain the necessary changes that you make to the HBRINST job. With these changes, you can quickly and easily

set up a new zRule Execution Server for z/OS that operates under the same console as an existing zRule Execution Server for z/OS.

Changes to HBRINST

When adding a Standalone zRule Execution Server for z/OS to an existing zRule Execution Server for z/OS console, you make a few changes (Table 8-13):

- You create a ++HBRSSID++ to start a new zRule Execution Server for z/OS instance.
- You create a ++HBRWORKPATH++ so that the logs, configuration data, res_data, res_xom, and work paths exist for each server.
- You create a ++HBRWORKDS++ to create the new changed working data sets.

Table 8-13 HBRINST customization values for rules on z/OS

Column one value	Your chosen value (column two)	Example value	Reason to update, change, or leave the default
++HBRSSID++		HBR1	Every time that a new zRule Execution Server for z/OS is set up, you must modify this value. So, setting a naming convention that can scale with your system is key. This value must be *four characters* or less.
++HBRWORKPATH++		/u/HBR1	This value is the work path for the specific instance of the server. You must update this value for each new zRule Execution Server for z/OS instance that you make. A preferred practice is to create a common directory for the work path directories. Then, create a directory within this common directory for each new instance.
++HBRWORKDS++		HBR.WORKDS.HBR1	For each new instance of the server, update this value to reference the new instance. This value is the data set name for the changed output from the HBRUUPTI PDSs.

Reusing a copy of the zRule Execution Server for z/OS HBRINST: If you use the preferred practice, you can reuse and update a copy of the existing zRule Execution Server for z/OS HBRINST if you deploy to the same zRule Execution Server for z/OS console. Using this approach, you make only the necessary updates.

Updating HBRUUPTI

Similar to the first configuration, you must update HBRUUPTI so that the new ++HBRWORKDS++ sets are created for the new instance of the Standalone zRule Execution Server for z/OS. Refer to 8.5.4, “Modifying and adding the started tasks to the PROCLIB” on page 253 for the procedure to update the corresponding lines in the JCL to create the new ++HBRWORKDS++ data sets. This procedure creates four new PDSs similar to the first instance, but with the new ++HBRWORKDS++ HLQ.

8.4.3 Creating the working directory

Next, you run the job HBRCRTI in the ++HBRWORKDS++ to create the working directory in the path that was set by ++HBRWORKPATH++. This job creates the UNIX System Services directory for the specific server instance.

8.4.4 DB2 persistence

The DB2 persistence layer that was set with the first Standalone zRule Execution Server for z/OS remains the same, so do not repeat this step.

8.4.5 Creating the new subsystem for the new Standalone zRule Execution Server for z/OS

The next step for the configuration of a new instance of the Standalone zRule Execution Server for z/OS is to define the subsystem in which the new instance runs. The systems programmer must perform this task. Figure 8-16 shows the task to run. ++HBRSSID++ is the subsystem ID that was set in Table 8-13 on page 248:

```
SETSSI ADD,SUBNAME=++HBRSSID++
```

```
                    System Command Extension

Type or complete typing a system command, then press Enter.

===> SETSSI ADD,SUBNAME=++HBRSSID++
===>
```

Figure 8-16 Setting the zRule Execution Server for z/OS instance subsystem name

8.4.6 Modifying and adding the started tasks to the PROCLIB

When adding an instance to an existing console, you create only one started task for this new instance, ++HBRSSID++MSTR. You copy HBRXMSTR as ++HBRSSID++MSTR to SYS1.PROCLIB or equivalent in your environment.

You must perform several steps to change the started task. You must make the DB2 persistent changes for HBRXMSTR.

Changes to HBRXMSTR

Change the following JCL in HBRXMSTR, which is in ++HBRWORKDS++.SHBRPROC(HBRXMSTR) from DSN=&HBRWORKD..SHBRPARM(HBRFILE) to DSN=&HBRWORKD..SHBRPARM(HBRDB2). Figure 8-17 on page 250 shows this change.

```
000020 //++HBRSSID++MSTR EXEC PGM=HBRMAIN,REGION=&REG
000021 //STEPLIB  DD DISP=SHR,DSN=&HBRHLQ..SHBRAUTH
000022 //SYSOUT   DD SYSOUT=&OUTCLAS
000023 //SYSABEND DD SYSOUT=&OUTCLAS
000024 //HBRLEOUT DD SYSOUT=&OUTCLAS
000025 //HBRPRINT DD SYSOUT=&OUTCLAS
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRMSTR)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRFILE)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-17 HBRXMSTR line to update from HBRFILE to HBRDB2 when adding DB2 persistence

Figure 8-18 shows the changed line.

```
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRMSTR)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRDB2)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-18 HBRXMSTR line updated to use DB2 persistence

At this time, you must change the data member called HBRMSTR, which is located in the ++HBRWORKDS++.SHBRPARM(HBRMSTR). In this data set member, change the HBRCONSOLE parameter:

```
HBRCONSOLE = NO
```

Figure 8-19 shows the change to NO.

```
000022 * Whether this Subsystem should start its own Console
000023 HBRCONSOLE=NO
000024
```

Figure 8-19 HBRCONSOLE to NO

If you do not change this data member to NO, when the instance is started later, it tries to start its own console. If the other instance is already running, this instance fails, because the ports are already in use.

Copy ++HBRWORKDS++.SHBRPROC(HBRXMSTR) to SYS1.PROCLIB as ++HBRSSID++MSTR. ++HBRSSID++ is the SSID of the new server instance that was created in Table 8-13 on page 248 for the specific instance.

SYS1.PROCLIB: SYS1.PROCLIB is the default. Change it to match the equivalent library in your environment. ++HBRSSID++ was set in Table 8-13.

Authorizing the server instance as a started task

Authorize the ++HBRSSID++MSTR as a started task procedure that executes with the same user ID. Use the following commands:

```
RDEFINE STARTED ++HJBRSSID++MSTR.* STDATA(USER(<HBRSSID_USER>)

GROUP(<HBRSSID_GROUP>)
```

8.4.7 Security setup for the new Standalone zRule Execution Server for z/OS

Complete the security setup for the new Standalone zRule Execution Server for z/OS by using the same method that is described in 8.3.3, “Securing the zRule Execution Server for z/OS for z/OS resources” on page 239. Refer to this section to perform the security setup for the RES server instance using the ++HBRSSID++ value that was set in Table 8-13 on page 248.

8.4.8 Starting the new instance

Similar to the first zRule Execution Server for z/OS instance, enter this command to start the new zRule Execution Server for z/OS instance:

```
START ++HBRSSID++MSTR
```

8.5 Configuring the zRule Execution Server for z/OS running on a CICS JVM server

Another new feature of WebSphere Operational Decision Management is the addition of a zRule Execution Server for z/OS that runs within the CICS JVM server on CICS V4.1 and CICS V4.2. The setup is similar for the two latest releases of CICS: V4.1 and V4.2. In this section, we describe the setup of this server instance. We create the configuration for running the zRule Execution Server for z/OS rule engine on a CICS JVM server. See Figure 8-20.

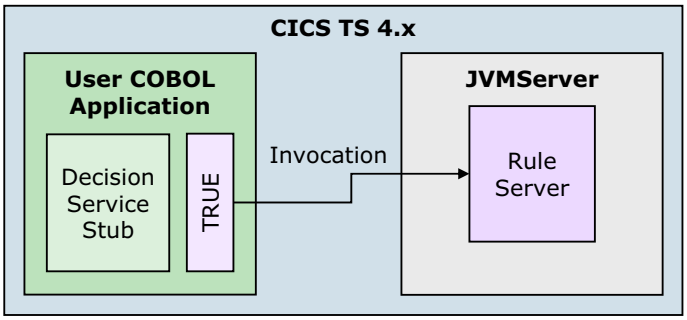


Figure 8-20 CICS COBOL application and JVM server

8.5.1 Updating HBRINST for the CICS zRule Execution Server for z/OS

You must update the HBRINST member for the CICS zRule Execution Server for z/OS to create a zRule Execution Server for z/OS running on a CICS server. Table 8-14 on page 252 shows the changes. After you make the changes, submit HBRUPPTI with the updated values for ++HBRWORKDS++ and HBRINST (if you followed the preferred practice to make the changes).

Table 8-14 HBRINST customization values for rules on z/OS

Column one value	Your chosen value (column two)	Example value	Reason to update, change, or leave the default
++HBRSSID++		HBR1	When creating a CICS zRule Execution Server for z/OS, you must modify this value. So, creating a naming convention that can scale with your system is key. This value must be *four characters* or less.
++HBRWORKPATH++		/u/HBR1	This value is the work path for the specific instance of the server. Update this value for each new zRule Execution Server for z/OS instance. A preferred practice for value is to create a common directory for the work path directories. Then, create a directory within this common directory for each new instance.
++HBRWORKDS++			For each new instance of the server, update this value to reference the new instance. This value is the data set name for the changed output from the HBRUUPTI PDSs.
++CICSWORKPATH++		/u/IG***	This value is the CICS UNIX System Services working directory path. For each server, we set this value so that each new zRule Execution Server for z/OS instance has its own CICS work path. Problem determination is easier by separating the logs for each zRule Execution Server for z/OS instance.
++CICSDSN++			The high-level qualifier for the CICS region system definition data set (CSD) file. For each new region into which a zRule Execution Server for z/OS is deployed, update this value.
++CICSLIST++			Name of the CICS list to which HBRGROUP is added. With each region change, update this value, if necessary.

8.5.2 Creating the working directories

For the CICS zRule Execution Server for z/OS, you must create two working path directories. The first task is the same task as configuring the Standalone zRule Execution Server for z/OS in that you submit the HBRCRTI job. The second job that you submit is the HBRCRCTI job,

which creates the CICS working path directories within UNIX System Services. Submit these jobs:

- HBRCRTI is the first job to submit. This job creates a directory at ++HBRWORKPATH++ for the logs, configuration files, and so on of the new server instance.
- HBRCRCTI is the second job to create. It creates the ++CICSWORKPATH++ for the CICS JVM zRule Execution Server for z/OS.

8.5.3 Defining a new subsystem for CICS JVM zRule Execution Server for z/OS

Updating the HBRINST data set makes a new ++HBRSSID++ for the CICS JVM server zRule Execution Server for z/OS. This new ++HBRSSID++ must be defined as a new subsystem in which the new instance runs. The systems programmer must run this task. Figure 8-21 shows the command to run, where ++HBRSSID++ is the subsystem ID that was set in Table 8-14 on page 252:

```
SETSSI ADD,SUBNAME=++HBRSSID++
```

```
                    System Command Extension

Type or complete typing a system command, then press Enter.

===> SETSSI ADD,SUBNAME=++HBRSSID++
===>
```

Figure 8-21 Setting the zRule Execution Server for z/OS instance subsystem name

8.5.4 Modifying and adding the started tasks to the PROCLIB

When adding an instance to an existing console, you need to create one started task for this new CICS JVM server zRule Execution Server for z/OS instance, which is ++HBRSSID++MSTR. You copy HBRXMSTR as ++HBRSSID++MSTR to SYS1.PROCLIB or the equivalent library in your environment.

You must make the DB2 persistent changes for HBRXMSTR.

Changing HBRXMSTR

Change the following JCL in HBRXMSTR, which is located in ++HBRWORKDS++.SHBRPROC(HBRXMSTR), from DSN=&HBRWORKD..SHBRPARM(HBRFILE) to DSN=&HBRWORKD..SHBRPARM(HBRDB2). See Figure 8-22 on page 254.

```
000020 //++HBRSSID++MSTR EXEC PGM=HBRMAIN,REGION=&REG
000021 //STEPLIB  DD DISP=SHR,DSN=&HBRHLQ..SHBRAUTH
000022 //SYSOUT   DD SYSOUT=&OUTCLAS
000023 //SYSABEND DD SYSOUT=&OUTCLAS
000024 //HBRLEOUT DD SYSOUT=&OUTCLAS
000025 //HBRPRINT DD SYSOUT=&OUTCLAS
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRMSTR)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRFILE)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-22 HBRXMSTR line to update from HBRFILE to HBRDB2 when adding DB2 persistence

Figure 8-23 shows the changed line.

```
000026 //HBRENVPR DD DISP=SHR,
000027 //            DSN=&HBRWORKD..SHBRPARM(HBRMSTR)
000028 //         DD DISP=SHR,
000029 //            DSN=&HBRWORKD..SHBRPARM(HBRDB2)
000030 //         DD DISP=SHR,
000031 //            DSN=&HBRWORKD..SHBRPARM(HBRCMMN)
```

Figure 8-23 HBRXMSTR line updated to use DB2 persistence

At this time, change the data member called HBRMSTR, which is located in ++HBRWORKDS++.SHBRPARM(HBRMSTR). In this data set member, change the HRRCONSOLE parameter:

```
HBRCONSOLE = NO
```

Figure 8-24 shows this change.

```
000022 * Whether this Subsystem should start its own Console
000023 HBRCONSOLE=NO
000024
```

Figure 8-24 HBRCONSOLE to NO

If you do not change HBRCONSOLE to NO, when you start the instance, it tries to start its own console. If the other instance is already running, starting this instance fails, because the ports are already in use.

Copy the ++HBRWORKDS++.SHBRPROC(HBRXMSTR) to SYS1.PROCLIB as ++HBRSSID++MSTR. ++HBRSSID++ is the SSID of the new server instance that was created in Table 8-14 on page 252 for the specific instance.

SYS1.PROCLIB: SYS1.PROCLIB is the default library. Change it to match the equivalent library in your environment. ++HBRSSID++ was set in Table 8-14.

Authorizing the server instance as a started task

Authorize the ++HBRSSID++MSTR as a started task procedure that executes with the same user ID. Use the following commands:

```
RDEFINE STARTED ++HBRSSID++MSTR.* STDATA(USER(<HBRSSID_USER>)

GROUP(<HBRSSID_GROUP>)
```

8.5.5 Creating the JVM profile

Next, create the JVM server profile for the version of CICS in which you set up your zRule Execution Server for z/OS. This step depends on the version of JVM server: either 31-bit on CICS V4.1 and 64-bit on CICS V4.2. To create the correct profile, submit one of these jobs:

- CICS V4.2: Run job HBRCJVMP to create the JVM profile for a CICS V4.2 region.
- CICS V4.1: Run job HBRCJS41 to create the JVM profile for a CICS V4.1 region.

This job creates the profile within the ++HBRCICSWORKPATH++ directory, which you must copy to your JVM profiles directory for your CICS region.

> **Important:** On CICS V4.2, the Java version that is included with the installation *must* be at a minimum Java V6.0.1. There is a version check within the WebSphere Operational Decision Management product that prevents the region from starting if the Java version is earlier than the recommended version.

8.5.6 Defining the CICS resources

Next, define the CICS resources that are required by the server. Submit two jobs that relate to the resources. The first job is the same job for both CICS V4.1 and CICS V4.2:

- HBRCSD: This job defines the resources that are required by both CICS V4.1 and V4.2.

The next step depends on the CICS version:

- HBRCSDJ: This job defines the necessary resources for a CICS V4.2 region.
- HBRCSD41: This job defines the necessary resources for a CICS V4.1 region.

8.5.7 Adding HBRLIST to the system initialization table

After defining the resources, add HBRLIST to the system initialization table that is specified by the GRPLIST parameter.

8.5.8 Changing the CICS region JCL

Next, modify the CICS V4.*x* region JCL to include the lines calling to the zRule Execution Server for z/OS on the CICS JVM. You must add the following lines to the components of the CICS JCL.

CICS STEPLIB concatenation

Add the SHBRAUTH PDS to the CICS region STEPLIB concatenation. Add the following line:

```
//        DD DSN=++HBRHLQ++.SHBRAUTH,DISP=SHR
```

DFHRPL

In the CICS program library, DFHRPL, add the SHBRCICS PDS to the DFHRPL section:

```
//        DD DSN=++HBRHLQ++.SHBRCICS,DISP=SHR
```

Passing the runtime variables to the CICS region

Last, you add the following runtime variables from the SHBRPARM PDS to the CICS region. The following two lines are required if you are running either database-based rules or file system-based rules.

For database persistence:

```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCICSJ)
//         DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCMMN)
//         DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCICSD)
```

For file system persistence:

```
//HBRENVPR DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCICSJ)
//         DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRCMMN)
//         DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRFILE)
```

If setting up the database environment, use the SHBRPARM(HBRCICSD) member. Also, make sure that the STEPLIB of the region contains the members SDSNLOAD and SDSNLOD2.

If setting up the file system environment, use SHBRPARM(HBRFILE) member.

Scenario for installation verification

If you plan to add the installation verification procedure (IVP) to test the zRule Execution Server for z/OS on CICS JVM server, add the following line in the runtime variables section:

```
//SCENARIO DD DISP=SHR,DSN=++HBRWORKDS++.SHBRPARM(HBRSCEN)
```

After the configuration is complete and the region is restarted later, this line enables the CICS MINI transaction (Miniloan sample application). MINI verifies that the rule engine is connected and working.

If running the IVP, you must submit the job for deploying the rule artifacts. For the database-based setup, run the job ++HBRWORKDS++.SHBRJCL(HBRDEPDB). If running a file system-based setup, submit the job ++HRBWORKDS++.SHBRJCL(HBRDEPFI). These jobs deploy the rule artifacts to be used by the zRule Execution Server for z/OS on the CICS JVM server.

8.5.9 Security for the zRule Execution Server for z/OS on CICS JVM server

For the security setup for the zRule Execution Server for z/OS on the CICS JVM server, perform the steps in 8.3.3, “Securing the zRule Execution Server for z/OS for z/OS resources” on page 239. You perform the same steps for all zRule Execution Server for z/OS servers.

8.5.10 Starting the zRule Execution Server for z/OS on CICS JVM server instance

After configuring and setting up the zRule Execution Server for z/OS instance on the CICS JVM server, start the zRule Execution Server for z/OS instance and restart the CICS region.

To start the server, run the following command:

```
START ++HBRSSID++MSTR
```

Then, start your CICS region.

8.5.11 CEDA installation of HBRGROUP resources

After starting the CICS region, you must install the resources that were defined earlier. Run the following command in CICS:

```
CEDA INSTALL GROUP(HBRGROUP)
```

8.5.12 Database connect for the CICS region

If you use the database configuration, perform the following step to connect the database to the CICS region of the zRule Execution Server for z/OS:

```
CEMT INQUIRE DB2CONN
```

Then, change the CONNECTST property from `Notconnected` to `Connected`.

8.5.13 Connecting the zRule Execution Server for z/OS to the JVM server

Next, connect the zRule Execution Server for z/OS to the JVM server. After the started task is up and the resources are connected, run the CICS transaction HBRC. If successful, this transaction returns the following message:

```
GBRZC9001I RC=0000
```

If the connection is unsuccessful, it returns `GBRZC9001E RC=`*`XXXX`*, where *`XXXX`* is the return code message. If it is a `3006` message, you did not start the ++HBRSSID++MSTR task yet. If it is a `3014` message, you did not enable the JVM server yet.

For the rest of the return codes, refer to the information center:

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/index.jsp?topic=%2Fcom.ibm.dserver.z.reference%2Fhtml%2Freasoncodes%2Fhtml%2Fcodes_zres.html&resultof=%223014%22

8.5.14 Deploying the installation verification program

If you set up the CICS region to have HBRSCEN, perform the following steps to run the installation verification program (IVP).

Deploying the RuleApp

Deploying the RuleApp depends on the persistence type that you set up. For database persistence, run the job HBRDEPDB. If the persistence is file system-based, run the job HBRDEPFI.

Running the installation verification program transaction

After you deploy the RuleApp, go back to the CICS region and run the CICS transaction MINI. Your region then displays, as shown in Example 8-2 on page 258.

Example 8-2 MINI output

```
MINICICS --connecting to Rule Execution Server
MINICICS --Loan customer 0000000001
MINICICS --about to call Rule Execution Server
MINICICS--name-Joe                          loan amount-005000000 -approved-F
MINICICS--msg-The loan cannot exceed 1,000,000
MINICICS --Loan customer 0000000002
MINICICS --about to call Rule Execution Server
MINICICS--name-Joe                          loan amount-000250000 -approved-T
MINICICS--msg-The loan amount is OK
MINICICS--msg-The borrower income is OK compared to the loan amount
MINICICS--msg-The credit score is OK
MINICICS --Disconnect from Rule Execution Server
MINICICS --SUCCESSFUL COMPLETION of demo
```

The setup of the zRule Execution Server for z/OS on the CICS JVM server is complete.

A

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

`ftp://www.redbooks.ibm.com/redbooks/SG248014`

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248014.

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material `.zip` file into this folder.

© Copyright IBM Corp. 2012. All rights reserved.

Abbreviations and acronyms

BAL	Business Action Language
BEP	Business Event Processing
BOM	Business object model
BPM	Business Process Management
DVS	Decision Validation Services
HLQ	High-level qualifier
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
IVP	Installation verification procedure
JMS	Java Message Service
JVM	Java virtual machine
KPI	Key performance indicator
RES	Rule Execution Server
SSP	Scenario service provider
XU	Execution unit
zFS	z/OS Distributed File Service

© Copyright IBM Corp. 2012. All rights reserved.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that publications referenced in this list might be available in softcopy only.

- *Implementing Event Processing with CICS*, SG24-7792
- *Proven Practices for Enhancing Performance: A Q & A for IBM WebSphere ILOG BRMS 7.1*, REDP-4775
- *Making Better Decisions using WebSphere Operational Decision Management*, REDP-4836

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- Barbara Hale, *Business Rules Applied*, Wiley, 2001, ISBN 978-0471412939

Online resources

These websites are also relevant as further information sources:

- WebSphere Operational Decision Management Information Center:

  http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.reference.doc/doc/webspherebusinesseventsprojects.html

- WebSphere Operational Decision Management documentation center:

  http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/index.jsp?topic=%2Fcom.ibm.dcenter.bu.rules%2FContent%2FBusiness_Rules%2F_pubskel%2FInfocenter_Primary%2Fps_DCBU_DCenter_Rules2495.html

- CICS Explorer website:

  http://www-01.ibm.com/software/htp/cics/explorer/

- To enable history in the Decision Server Events run time, follow the instructions documented at this website:

  http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.install.doc/doc/configuringwebspherebusinesseventsruntimetorecordhistory.html

- For information about the WebSphere connectors, go to this website:

© Copyright IBM Corp. 2012. All rights reserved.

http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.admin.doc/doc/runningconnectorsinthewasenvironment.html

- Go to this website for information about the stand-alone connectors:

  http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.wbe.admin.doc/doc/zos_runningtechnologyconnectors.html

- Package ilog.rules.dvs.client extensive online documentation with code samples that explain how to use it:

  http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/topic/com.ibm.dserver.reference.studio/html/api/html/ilog/rules/dvs/client/package-summary.html

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Flexible Decision Automation for Your zEnterprise with Business Rules and Events

Understand the benefits of operational decision management

Build dynamic solutions with business events and business rules

Learn by example with practical scenarios

The IBM WebSphere Operational Decision Management product family provides value to organizations that want to improve the responsiveness and precision of automated decisions. This decision management platform on IBM z/OS provides comprehensive automation and governance of operational decisions that are made within mainframe applications. These decisions can be shared with other cross-platform applications, providing true enterprise decision management.

This IBM Redbooks publication makes the case for using WebSphere Operational Decision Management for z/OS and provides an overview of its components. It is aimed at IT architects, enterprise architects, and development managers looking to build rule-based and business event-based solutions. We provide step-by-step guidance on getting started with business rules, and in creating business events, taking a scenario-based approach. This book provides detailed guidelines for testing and simulation, and it describes advanced options for decision authoring. Finally, we describe and document multiple runtime configuration options.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8014-00 ISBN 073843678X